Calhoun: The NPS Institutional Archive

Theses and Dissertations | Thesis Collection

2002-09

Probabilistic modeling and simulation of metal fatigue life prediction

Heffern, Thomas V.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/5098

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PROBABALISTIC MODELING AND SIMULATION OF METAL FATIGUE LIFE PREDICTION

by

Thomas Vernon Heffern
September 2002

Thesis Advisor: Ramesh Kolar
Second Reader: E. Roberts Wood

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Probabilistic Modeling and Simulation of Metal Fatigue Life Prediction			**5. FUNDING NUMBERS**
6. AUTHOR(S) Major Thomas V. Heffern			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

As fiscal constraints demand maximum utilization, engineers must develop more rigorous methods to predict the life limits of aircraft components. Current Navy policy requires that aircraft and aircraft parts be retired before they reach 100% FLE. An investigation has been initiated that would attempt to quantify the probability of failure if aircraft parts were extended in service life beyond 100% FLE.

The work of this thesis was to investigate the probability distributions of test data taken for aluminum 7050-T7451, and to attempt to develop a probability based model from the variation of the 4 fatigue life constants (σ_f' , ε_f' ,b,c). The goal was to create strain-life-probability curves that would more accurately describe the likelihood of failure at a given strain amplitude.

The investigator determined that the test data did not demonstrate any consistent known probability density function. The investigator cautioned against assuming a normal distribution before it could be completely established as the predominate probability density function. Possible consequences of invalid assumptions were presented. Attempts were made to explain the disparity of sample data between two different laboratories testing of the same material.

Assuming random behavior within an established range, probability based models were developed using the 4 strain-life constants. It was determined that in order to create a complete probability based model, an accurate regression of the test data must fit all strain levels to include the intermediate strain level's "knee". In an attempt to solve that problem, 8 parameter equations were explored. Methods to predict the 8 parameters included random number simulation combined with non-linear least squares curve fits, evolutionary algorithms and genetic algorithms.

14. SUBJECT TERMS Metal Fatigue, Variable Strain-Life, Coffin-Manson, Monte Carlo Simulation, Service Life Extension, Evolutionary Algorithms, Genetic Algorithms, Aluminum 7050-T7451, Probabilistic Investigation			**15. NUMBER OF PAGES** 134
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PROBABILISTIC MODELING AND SIMULATION OF METAL FATIGUE LIFE PREDICTION

Thomas V. Heffern
Major, United States Marine Corps
B.S., Mechanical Engineering, Virginia Military Institute, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author:

Thomas V. Heffern

Approved by:

Ramesh Kolar
Thesis Advisor

E. Roberts Wood
Second Reader/Co-Advisor

Maximilian Platzer
Chairman, Department of Aeronautics and Astronautics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As fiscal constraints demand maximum utilization, engineers must develop more rigorous methods to predict the life limits of aircraft components. Current Navy policy requires that aircraft and aircraft parts be retired before they reach 100% FLE. An investigation has been initiated that would attempt to quantify the probability of failure if aircraft parts were extended in service life beyond 100% FLE.

The work of this thesis was to investigate the probability distributions of test data taken for aluminum 7050-T7451, and to attempt to develop a probability based model from the variation of the 4 fatigue life constants ($\sigma_f^{'}, \varepsilon_f^{'}$,b,c). The goal was to create strain-life-probability curves that would more accurately describe the likelihood of failure at a given strain amplitude.

The investigator determined that the test data did not demonstrate any consistent known probability density function. The investigator cautioned against assuming a normal distribution before it could be completely established as the predominate probability density function. Possible consequences of invalid assumptions were presented. Attempts were made to explain the disparity of sample data between two different laboratories testing of the same material.

Assuming random behavior within an established range, probability based models were developed using the 4 strain-life constants. It was determined that in order to create a complete probability based model, an accurate regression of the test data must fit all strain levels to include the intermediate strain level's "knee". In an attempt to solve that problem, 8 parameter equations were explored. Methods to predict the 8 parameters included random number simulation combined with non-linear least squares curve fits, evolutionary algorithms and genetic algorithms.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis would not have been completed had it not been for my exceedingly devoted and enduring wife. She drove her pregnant self, 3 toddlers, a Dalmatian, a cat, and a hamster, (as well as 13 cases of California wine!) from Monterey, California to Patuxent River Maryland, 4 months before this investigator was complete with his thesis work! This enabled a level of dedication to this paper that would have been otherwise unattainable. Certainly, she is the epitome of what a Marine Wife should be.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. METAL FATIGUE

The prediction of the useful life of metals has been of interest to man since the advent of the railroad. How long a metal will withstand the level of applied stress presents many interesting problems. Of interest to this paper is that there is a wide variation in how long a given metal will last at a constant level of stress or strain. Simply put, no two samples, even though manufactured exactly the same will ever have exactly the same physical constitution. Therefore, their lives will not be the same. Obviously, the more man is able to control and refine his processes, the less the materials will vary. Until the day when a process has been created to make each piece exactly the same to include the orientation of each molecule, there will always be variation to deal with. Probabilities, statistics, and variational analysis are just a few of the ways people attempt to explain the unknown. This paper takes a look at how the variation of each metal may be better quantified and possibly predicted.

B. STRAIN-LIFE

(The following discussion on Strain-Life relies heavily on Fundamentals of Metal Fatigue Analysis [1].)

For years, engineers and mathematicians have quantified metal fatigue with the use of sample testing. By testing small coupons of a given material, according to American Society of Testing Materials (ASTM) methods, engineers predicted the average life a material would have at a given stress or strain level. The results were typically depicted in the form of stress–cycles to failure or strain–cycles to failure. Traditionally, samples were tested with constant amplitude loadings. The heart of this analysis was the following relationship between stress and strain:

$$\sigma_{stress} = E\varepsilon_{elastic}$$

This equation describes the interaction between stresses applied to a material and the percent deformation due to those stresses (strain). The two are related by the elastic

properties of the material, the Modulus of Elasticity (E). Upon loading, the material's deformation is resisted by the modulus of elasticity. Once the load is no longer applied, the material "snaps" back to its original size with no permanent deformation. When a material is never stressed to the point at which it permanently deforms, the material is said to be undergoing elastic strains. When this type of behavior is applied to alternating or cyclic stresses, it is describe as High Cycle Fatigue, since the material life-times associated with non-deforming load levels are much higher than materials subjected to loadings where there is permanent deformation. High Cycle Fatigue (HCF) is thus usually tested with constant amplitude load test. When the material does fail, the determination of the level of strain is a simple transformation:

$$\varepsilon_{elastic} = \sigma_{applied} \div E$$

During testing, this equation is used to evaluate the strain at failure based on an established constant amplitude load level. At each failure, the number of cycles to failure (2Nf) is recorded. (A cycle is counted as, zero stress, to maximum compressive stress, to maximum tensile stress, and then back to zero load.) After many samples have been tested at different load levels the failure points are plotted on log-log paper, and predictions can be made about the life of the material based on a given stress level applied. The life is determined from the following equation:

$$\frac{\Delta\sigma}{2} = \sigma_f' \times 2Nf^b$$

$\frac{\Delta\sigma}{2}$ = stress amplitude

σ_f' = fatigue strength coefficient

b = fatigue strength exponent

$2Nf$ = cycles to failure

In log space, a lined describes the trend of the data points. The fatigue strength coefficient is the Y intercept and the fatigue strength exponent is the slope of the line.

During HCF the elastic strain equals the total strain. If materials are subjected to larger loads, when deformation takes place, it includes two effects, elastic and plastic. With a large enough load applied, a material will not "snap" back into shape after the

load is released. Instead there will be some amount of permanent deformation. The relationship between strains is given as:

$$\varepsilon_{total} = \varepsilon_{elastic} + \varepsilon_{plastic}$$

When permanent deformations occur, the Stress-Life Methods can no longer be used to analyze the strain at failure. Strain-Life Methods must be used. It was realized that a similar log-linear transformation could be applied to plastic strain.

$$\frac{\Delta\varepsilon_{plastic}}{2} = \varepsilon_f^{'} \times 2Nf^{c}$$

$\frac{\Delta\varepsilon_{plastic}}{2}$ = plastic strain amplitude

$\varepsilon_f^{'}$ = fatigue ductility coefficient

c = fatigue ductility exponent

As with the Stress-life equation, the fatigue ductility coefficient represents the Y intercept and the fatigue ductility exponent represents the slope of the log-linear line.

If the Stress-life equation is converted to strain amplitude the following equation is derived:

$$\frac{\Delta\varepsilon_{elastic}}{2} = \frac{\sigma_f^{'}}{E} \times 2Nf^{b}$$

Since it has been previously determined that total strain is the combination of elastic and plastic strain, the two previous equations are combined to yield the Strain-life equation, also known as the Coffin-Manson equation:

$$\frac{\Delta\varepsilon_{total}}{2} = \frac{\sigma_f^{'}}{E} \times 2Nf^{b} + \varepsilon_f^{'} \times 2Nf^{c}$$

Since Strain-life includes the plastic effects, it is usually used when evaluating Low Cycle Fatigue. LCF, by definition, has a shorter life than HCF. This is due to the fact that larger loads are applied, plastic effects are introduced and the materials breaks much more quickly. Strain-life methods usually determine the required coefficients and exponents from constant amplitude strain tests. Since the strain-life method has wider

application, it is the chosen tool for the Navy's aircraft metal fatigue prediction. When samples are tested with this method, a variable load is applied that will result in a constant strain amplitude. Initially, the material exhibits only elastic strain. It is loaded, it deforms, and then it is unloaded and it snaps back into place. During this stage, while plastic deformation does not occur, the relationship between stress and strain is linear, related by the modulus of elasticity. Since the strain is constant, the load to produce that strain remains constant. At some point, as the material properties begin to break down, micro-cracks develop, permanent deformation occurs, and the load requirement to cause the established strain level is reduced. As the material becomes increasingly weakened, there is a significant decrease in load required to strain the material (load drop). When the material no longer resists the load, it breaks.

When samples are tested in a strain controlled manner, the resulting plot of the elastic strains regression line, added to the plastic strains regression line, yields the equation previously established as the Strain-life or Coffin-Manson equation. The following is an example of a number of tests with regression lines of the plastic and elastic strain levels. The sum of these two equations would predict the mean life of this material for a given constant amplitude strain level.

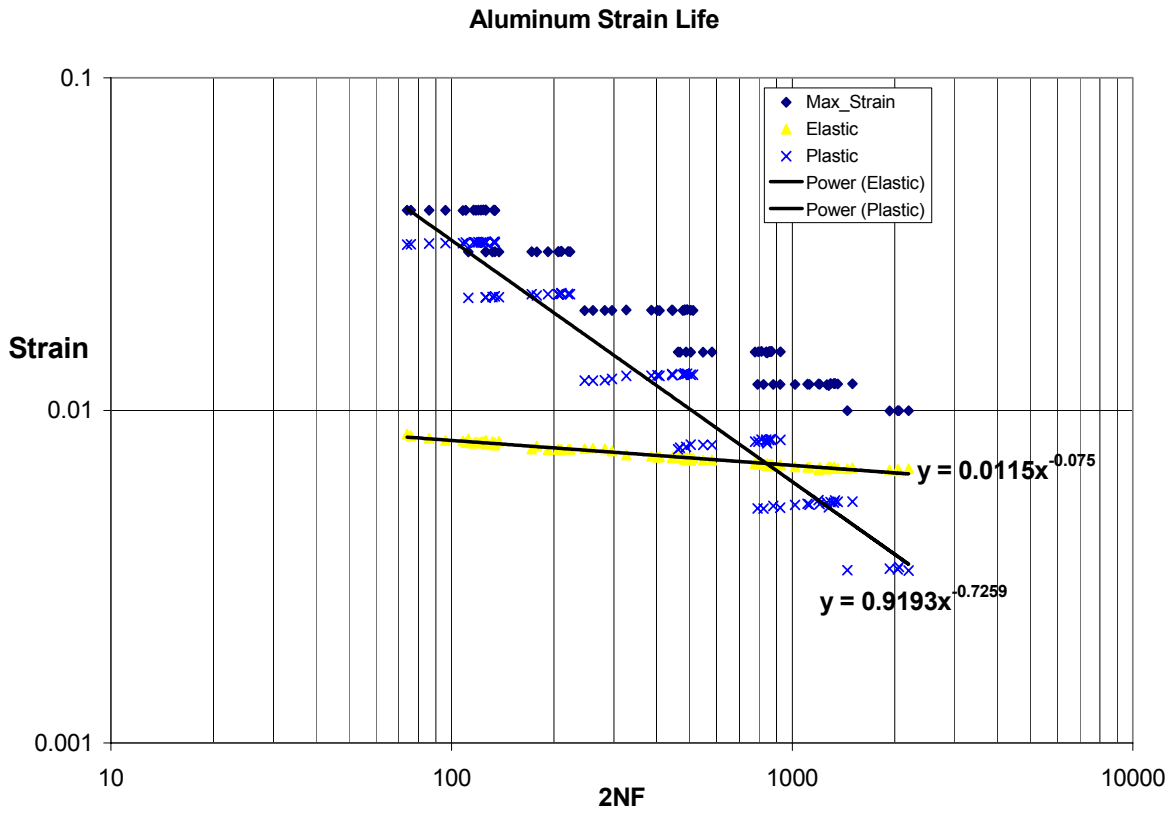

Figure. 1. Example Strain-Life Plot

In log space, a power law regression to the plastic and elastic points provides the basis for the Coffin-Manson equation [1].

$$\frac{\Delta\varepsilon}{2} = \frac{\sigma_f'}{E}(2NF)^{(b)} + \varepsilon_f'(2NF)^{(c)}$$

For this data set the equation becomes:

$$\frac{\Delta\varepsilon_{total}}{2} = .115\times(2Nf)^{(-.075)} + .913\times(2Nf)^{(-.7259)}$$

This equation defines a predicted average life at a given strain. Because this method is based on a regression, the equation does nothing to describe the variation that the material has. There is a variation of life, which is demonstrated by the scatter of the data points. The following figure demonstrates the strain-life equation, as it was derived from the regression of the elastic and plastic parts of this data set.

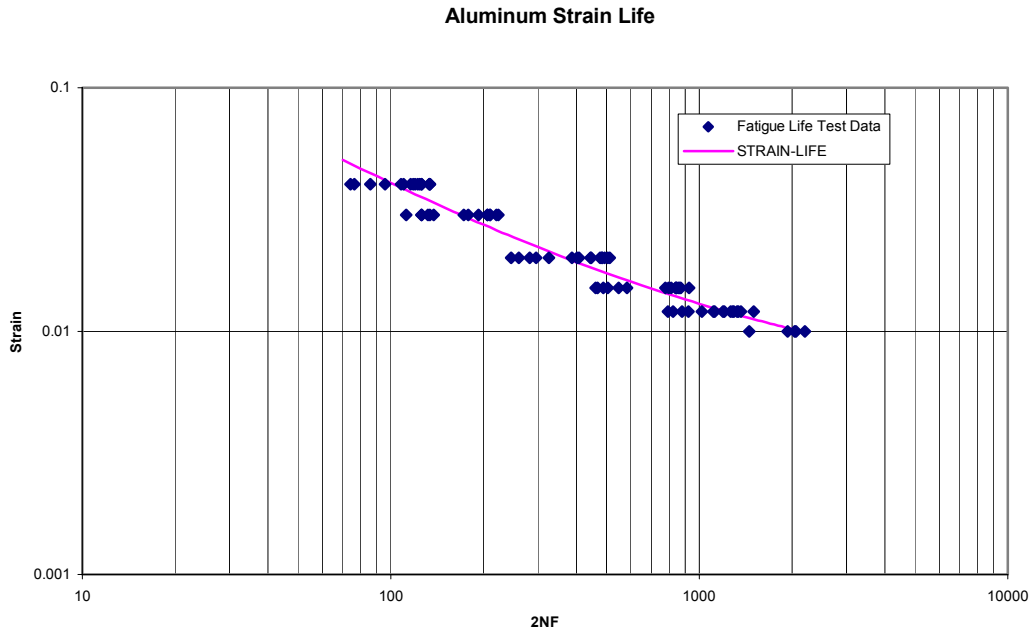

Figure. 2. Strain-Life Equation

C. PROBABALISTIC METHODS

To compensate for the fact that the strain-life equation only predicts an average life, engineers typically use a scatter factor to account for the variance that a material may have. Obviously, an aeronautical engineer would want to design for the worse possible case. For that matter, a scatter factor of 4 is often used to translate from the design stage to the actual product. This large scatter factor helps to account for actual material geometries, manufacturing defects and other uncertainties. This factor attempts to correlate a coupon test to a full-scale structure. Other less sensitive designs may be able to use scatter factors as low as 2 or 3.

Since the late 1940s, engineers have examined the test data in an attempt to define a probability distribution that would explain and characterize the behavior of the data. Early advances of this type of analysis, were popularized by the famed W. Weibull with his 1951 paper, "A Statistical Distribution Function of Wide Applicability" [2]. To date there are many different methods that attempt to predict the probabilities of failure. Some of the methods include, First/Second Order Reliability Methods (FORM/SORM), Monte Carlo simulation, and probabilistic finite element. [4]. Many involve the

intersection of the probability distribution function of failure of a given material with the probability distribution function of the load levels.

The goal of such probabilistic methods is to develop a more accurate prediction of fatigue life. This would ultimately increase the range of useable life over the scatter factor methods with an increased confidence of safety.

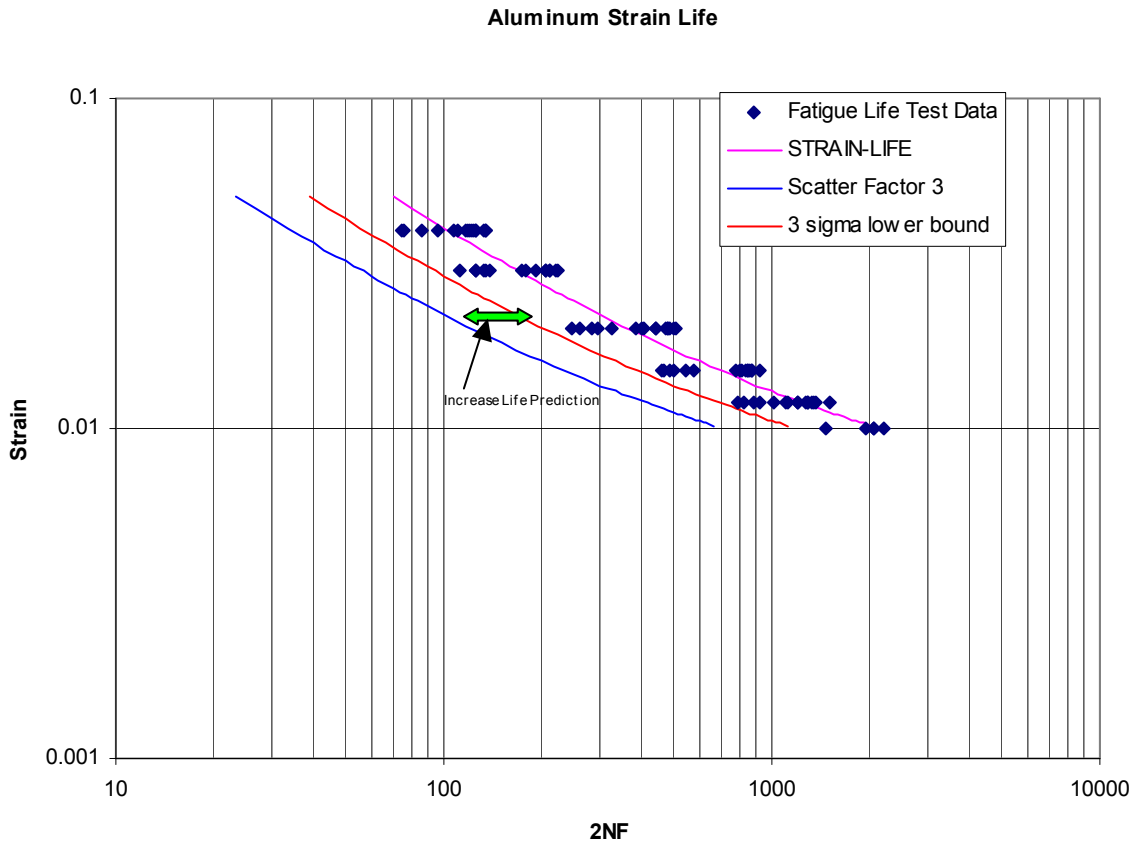

Figure. 3. Hypothetical Probabilistic Model

D. PROBLEMS WITH STRAIN-LIFE

In addition to the problem that the actual life at a given strain level cannot be predicted, with certainty, because of the scatter, there are other difficulties. There exists a significant problem with the strain-life equation at the transition from LCF to HCF in the "Intermediate" region. The actual data is not always exactly log-linear and there exists a

"knee". Additionally, there is often another "knee" in the high cycle elastic strain data. Figure (4) demonstrates these problems.

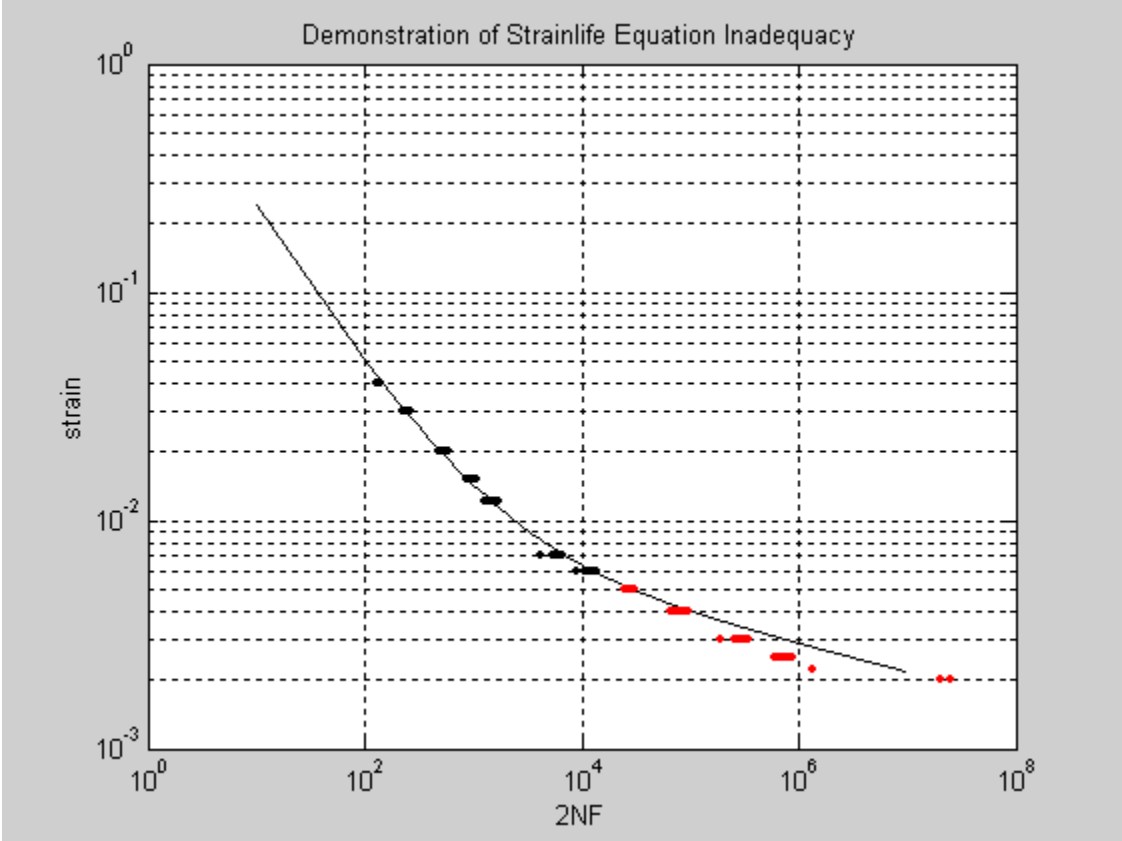

Figure. 4. Strain-Life Equation Problems

Note that the equation matches the first five levels very well, however, at level six, there exists an irregularity to the data. The last several high cycle levels also move away from the line. Most fatigue data does not have a "well behaved" and simple solution through all mean points. The data points are often piece wise continuous at best.

E. GOALS

The goal of this thesis was to investigate the possibility of predicting the probabilistic fatigue life of the material by varying the four strain-life fatigue constants (σ_f', ε_f',b,c) using Monte Carlo simulation. Genetic Algorithms were also to be investigated as a means of predicting the fatigue life. The objective was an efficient method that would use Monte Carlo simulation to make accurate predictions about fatigue life based on a small amount of test samples. The desire was the ability to take a

few parameters of the material properties, possibly from the monotonic (static) data, and to create a probabilistic strain-life model. A hope was that the simulation or genetic algorithm methods would somehow be able to correct for the "knee" problem. If the simulation could produce results that matched the test data, then strain life probability curves could be produced that would be able to more accurately specify the fatigue life of a material. A goal would be to be able to use minimal data and still be able to more safely and accurately predict the safe life region of material at different strain and ultimately stress levels.

In addition to a simulation model, there was a requirement to determine the best testing method to properly characterize the probabilistic nature of the material's fatigue life. The data used for this thesis was compiled from the NAVAIR structures division. Their data included an average of 15 data samples over 13 strain levels. NAVAIR testing methods will be explained in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ANALYSIS OF NAVAIR TEST DATA

A. NAVAIR GOALS

Aging aircraft are of increasing concern in aviation, especially Naval Aviation. NAVAIR's safe life methodology is the strain-life approach. Currently aircraft are retired from service before they reach 100% Fatigue Life Expended (FLE). FLE is defined as an aircraft having a 1 in 1000 chance of having a .0100 inch crack or larger in a structural member. As air platforms are increasingly extended in service to due to operational and fiscal requirements, there is an approaching time when aircraft may be required to operate beyond 100% FLE. While the FLE does provide a significant safety factor (1 in 1000 chance), an important question is, with what certainty can safe life be predicted beyond 100% FLE? In order to be able to address this question with confidence, the structures division of NAVAIR has undertaken the task of developing a probability-based strain life model. [3] (NOTE: A 1 in 1000 chance means a probability of .001. This also happens to correspond to the probability at -3 sigma on a standard normal probability curve.)

B. NAVAIR DATA

In order to develop the required database for a probability based strain-life model, an experimental strain-life test program was initiated by NAVAIR. The tests consisted of hourglass and uniform gage section test specimens. All specimens were cut from the same piece of sheet metal. The testing was accomplished in accordance with ASTM E606 for low cycle fatigue, and E466 for high cycle fatigue. Cycle times were investigated at 2%, 5%, 10% 15%, 20%, and rupture. NAVAIR selected the 10% load drop level to correspond to a .01 inch crack. This assumption was based on what other testing agencies have done in the past. It is not reflective of what the actual size of micro-cracks exist at 10% load drop. To date, there has not been a thorough analysis of this variable.

The investigator of this thesis took a slightly different approach. He considered that the "intent" of the NAVAIR definition of a 1 in 1000 chance of a .01 inch crack

really meant that a crack existed that would cause failure. The following chart demonstrates that at a 2% load drop, there must exist a crack of unknown size that is rapidly leading to metal failure. In theory, without a crack, fatigue life is infinite. Therefore, a 2% load drop demonstrates that there is a crack and the material is going to break shortly. In the following chart, each line represents a sample's fatigue life as a function of its load drop during the crack initiation and failure process. The X axis is the % load drop, so as the load drop goes from 0 to 1 (100%load drop), the life time of the material is plotted on the Y axis. As an example, in the top cluster of samples, when the load first begins to drop (2%), there has been approximately 8000 cycles. As the load required to strain the material drops even further to 5%, the material has lasted about 9500 cycles. When the load finally drops by 100% and the coupon breaks in two, it has lasted approximately 10,000 cycles.

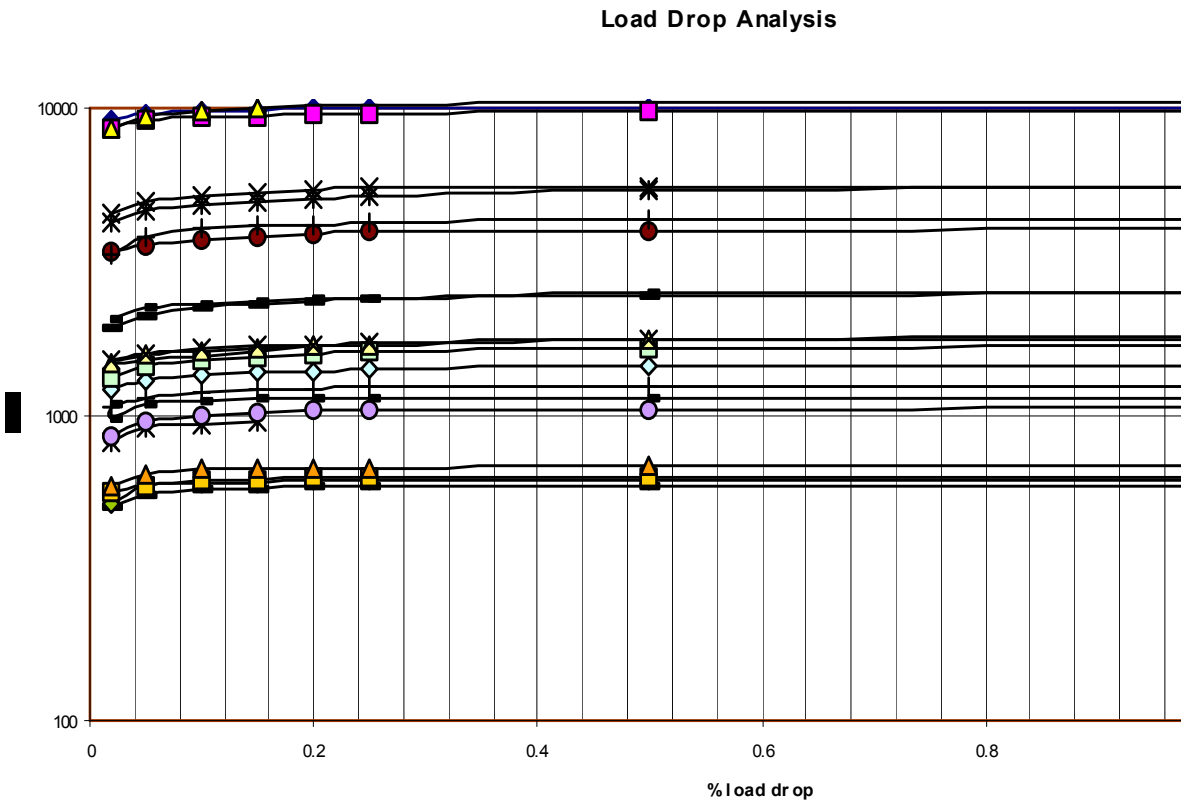

Figure. 5. Load Drop Analysis of Aluminum 7050-T7451

It can readily be seen from figure (5) that at a 2% load drop, there must exist a crack, and the material is quickly headed to failure. At 10% there also certainly exist a crack. The problem with using 10% is that from 10% to 100% (failure), there is little fatigue life left. The life at 10% is almost the same as life at failure. Therefore, the author assumed, that 10% essentially defined failure. If the intent of the NAVAIR definition actually was meant to serve as an indicator of a 1 in 1000 chance of failure, then clearly, 10% would be a satisfactory pick. However, if the intent is that just a crack exists, then 2% should be used. In order to quantify failure as a .01 inch crack, the author selected the cycle count at a 2% load drop to define the end of useful life. For more information about this aspect of the test sequence see ref [3].

C. EXTRACTION OF NAVAIR DATA

The NAVAIR investigation team, tested AL 7050-T7451 in accordance with reference [3] . The NAVAIR data was a compilation of samples tested in their own laboratories and samples tested under contract by METCUT laboratories. It was determined during the course of their testing that the grain orientation of the samples could produce significant testing errors. For this reason, the majority of the samples tested by the NAVAIR laboratories were rejected since they were not grain controlled. The testing error was realized before the testing by METCUT laboratories. Therefore, the majority of the samples investigated in this thesis were actually tested by the METCUT laboratory. The interested (or not) reader can learn more about the grain control error from reference [3].

The results of all samples (METCUT and NAVAIR, Grain controlled and uncontrolled, Uniform and Hourglass) were recorded to a Microsoft Access database. This information was provided to the Naval Post Graduate School. The investigator then queried the material for grain-controlled, 2% load drop data and arranged the data for Microsoft Excel. The following strain levels were selected for analysis: METCUT - .04 (16 data points), .03(15 data points),.02(16 data points),.015(16 data points),.012(16 data points),.010(05 data points),.008(5 data points),.007(15 data points),.006(15 data points); NAVAIR - .005(15 data points), .004(15 data points), .003(15 data points), .025(10 data

points), .022(1 data point), .002(5 data points). The 2% grain-controlled LCF METCUT data and the HCF NAVAIR data is provided in Appendix A.

The following chart presents the test data from Appendix A, which was also used to analyze the probability distributions and to create the strain-life probability models.

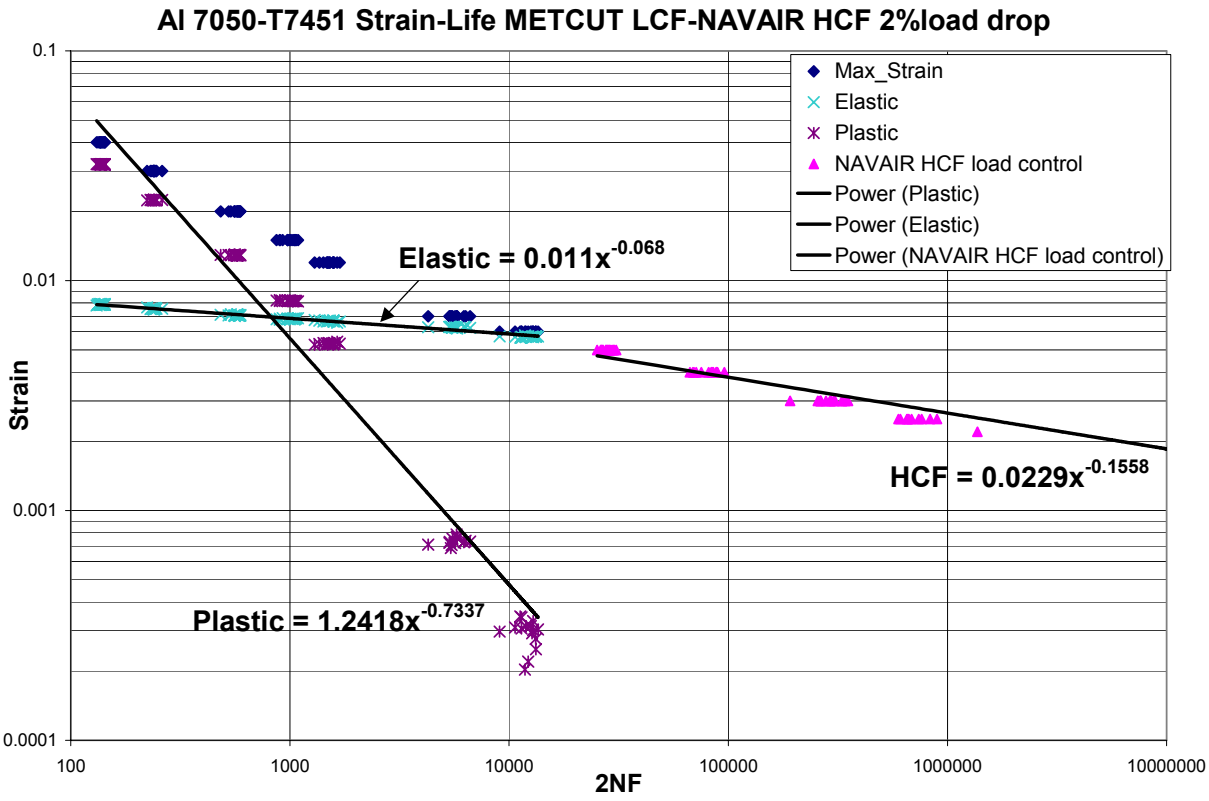

Figure. 6. Strain-Life Plot

Since only 5 data points exist for the NAVAIR data, only the METCUT data was used for a power-law regression of the elastic and plastic data points. The coefficients and the exponents provide the solution to the standard Coffin-Manson Strain-Life Equation.

$$\frac{\Delta\varepsilon}{2} = .011\times(2Nf)^{(-.068)} + 1.2418\times(2Nf)^{(-.7337)}$$

Since the regression line of the NAVAIR HCF data demonstrates a different slope and intercept than the METCUT elastic regression line, the parameters of the NAVAIR HCF regression line were used as an adjustment for different models proposed by this paper.

D. AL 7050-T7451 STATISTICAL ANALYSIS

Most natural phenomenon can be best described by a normal type distribution according to the Central Limit Theorem. This is generally accepted to be the case with fatigue data, though sometimes it is more accurately characterized by the lognormal or Weibull distribution. Initially, in the development of the probabilistic model proposed, a normal distribution of the experimental data was assumed. This followed the assumptions of the NAVAIR probabilistic model [3]. Evaluating the data with normal probability plots, the results look reasonably normal since the data points appear to be described by a straight line.

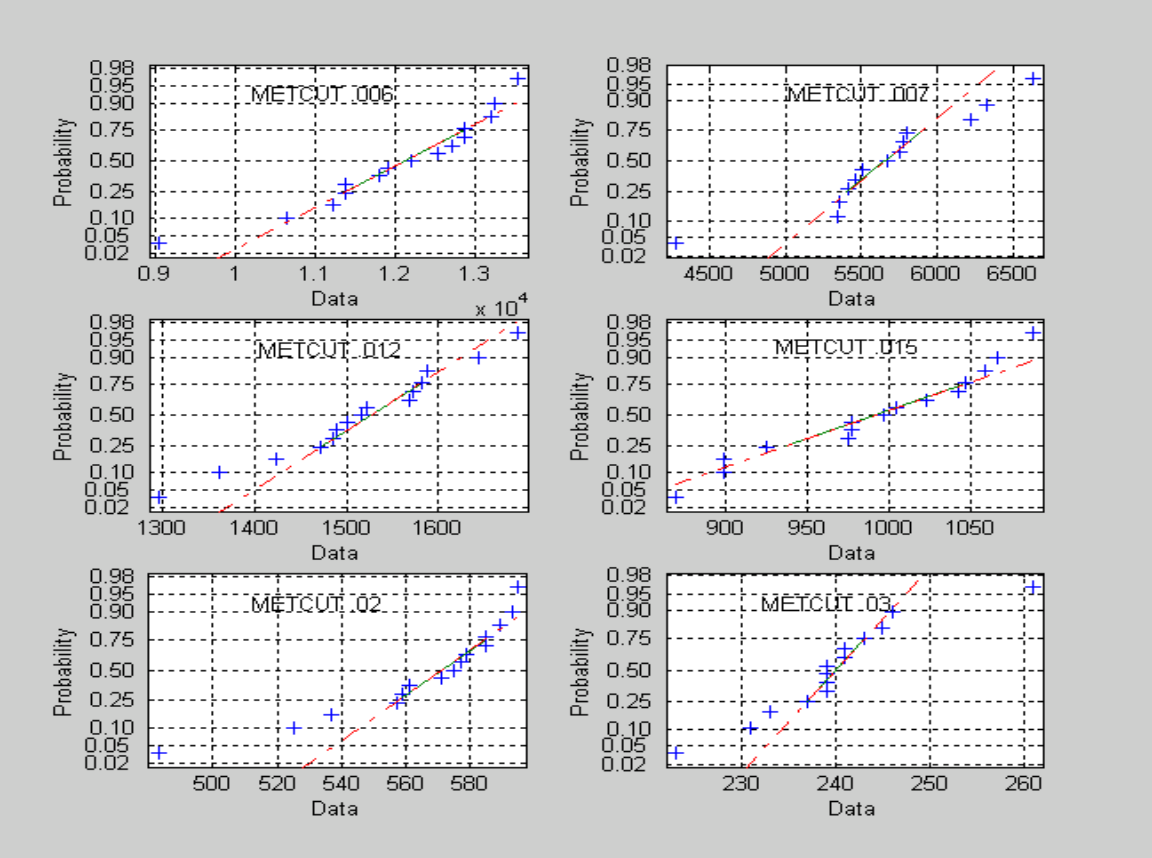

Figure. 7. METCUT Data Normal Probability Plots

Upon a closer analysis, the aluminum test data was not found to correlate well to a normal distribution. Visually, several strain levels appear to be moving toward a normal distribution. However, each strain level seems to exhibit a different form of behavior. A very distinctive normal behavior is not seen.

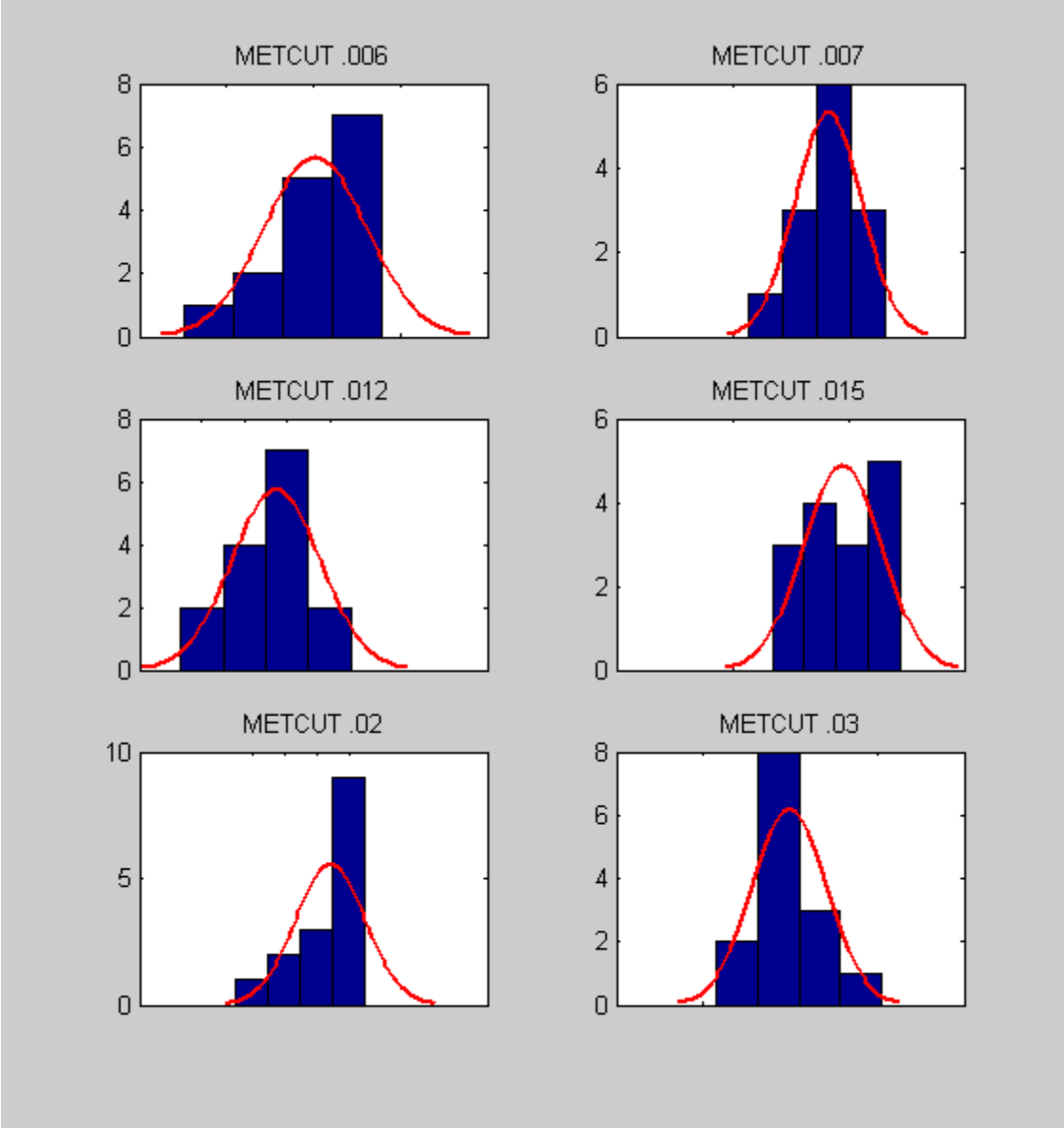

Figure. 8. Test Data Frequency Plot with Superimposed Normal PDF

A visual inspection of the frequency plots of the data points paints the most accurate picture of what the data represents. However, in the interest of accuracy, Crystalball® software was used to investigate the best fit of the data. Crystalball® was used to evaluate the test data with the Chi squared test, the Anderson- Darling Test and the Kolmogorov-Smirnov Test in order to evaluate the best fit to 11 standard distributions (Normal, Lognormal, Weibull, Uniform, Logistic, Extreme Value, Pareto, Gamma, Beta,

Exponential, and Triangular). The Kolmogorov-Smirnov Test was chosen since there were too few points for a Chi-squared evaluation and the Anderson-Darling is for use when there is concern at the "tails" of the distribution. The following best fits were found with the 2% Load Drop, Grain-controlled METCUT data and the NAVAIR HCF Grain-controlled data:

Table 1. Al 7050-T7451 Test Data Distribution Best Fits

Strain Level	Best Fit Probability Density Funtion	Kolmogorov-Smirnov coeff
.004	Extreme Value	.1359
.005	Triangular	.0838
.006	Weibull	.0933
.007	Logistic	.1845
.012	Logistic	.113
.015	Beta	.1066
.02	Beta	.1066
.03	Normal	.1776
.04	Gamma	.1416

The statistical analysis demonstrates that the data points lack the measures of standard normal distribution and there is no real trend between various strain levels. Additionally, the Kolmogorov-Smirnov correlation coefficient should typically be less than .03 in order to identify a good fit. No such "good fit" was found with the present test data. Altrnatively, the data could be described by a random distribution with a "few more" data points scattered about the middle of the distribution. It should also be noted that the investigator conducted similar investigations of the 5% and 10% load drop data and found the characteristics of the probability distribution to be just as varied as the 2% data set.

A heuristic estimate can be made for the number of samples that will be required to demonstrate a normal distribution. Assuming that the final profile of an infinite group of samples does demonstrate a normal distribution, then a visual inspection of figure (8) reveals that in all cases presented, there are several data points lacking in the 2 to 3 sigma

area of an imagined normal curve. The MATLAB simulated normal curve (in red) helps with this visualization. In order to develop a normal curve about the apparent mean point of the data in each set, there must exist approximately 4 more data points in the region between 2 and 3 standard deviations. The probability of obtaining points within this region is 15.87%. Since, (25 X .1587 = 4), 25 more data points are required by testing to validate a normal distribution and populate the region of deficiency. The following figure visually describes the estimate.

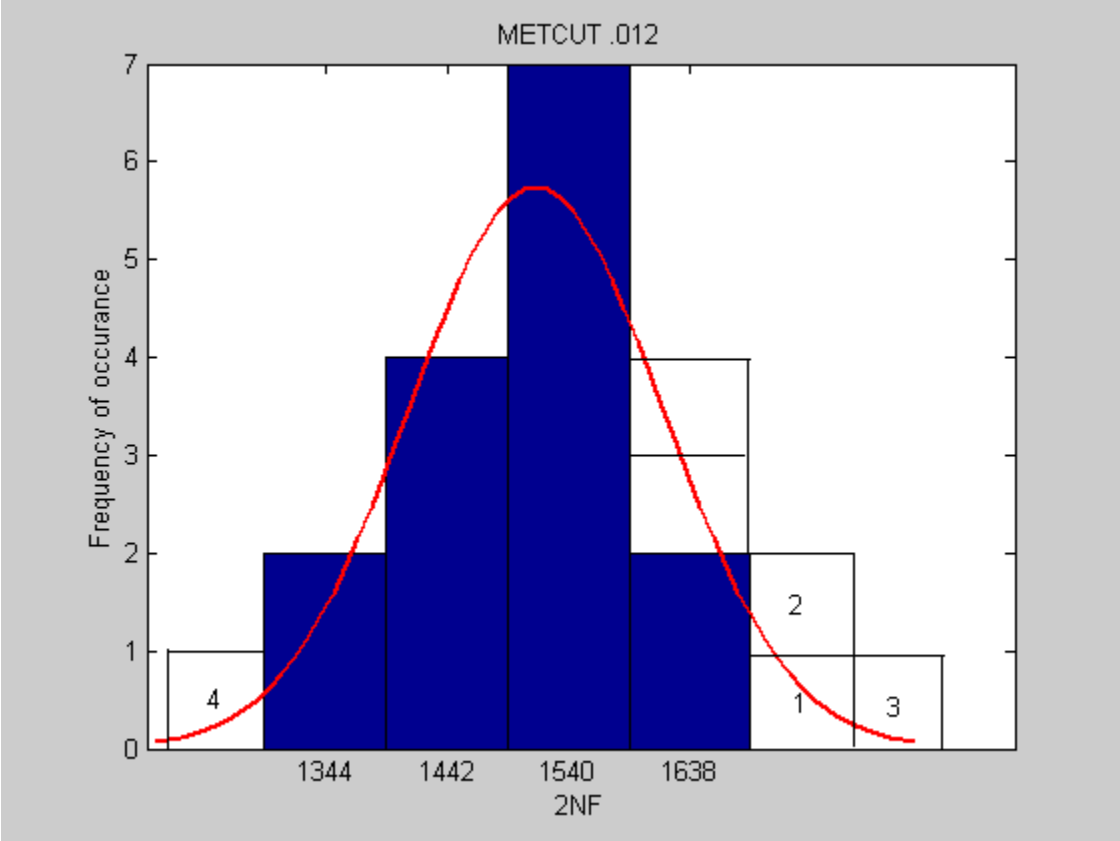

Figure. 9. Estimate Method for Number of Samples Required for Normal Distribution

E. AN ATTEMPT TO DEMONSTRATE A NORMAL TREND

A method has been presented to represent the test data in a way to understand the possible trends and behavior of the distribution. In an attempt to gain some indication of the type of trend to the data, the test points were all normalized between 0 and 1 at each strain level. The reason for taking such measures was to compensate for the lack of data points at the given strain levels. With this approach all test points could be plotted together. Normalization was accomplished in the following manner. Step 1: Subtract all data points in a given level by the smallest 2Nf value. This moved each range to the zero axis. Step 2: The same points (having been subtracted) were again divided by the largest new 2Nf value of the new range. This had the effect of making all the data points within a given level plot lie between 0 and 1. Thus all strain level data points were put on the same " playing field". The following chart presents this normalized data with the normalized 2Nf on the X axis, and the frequency (number within a set range) on the Y axis.

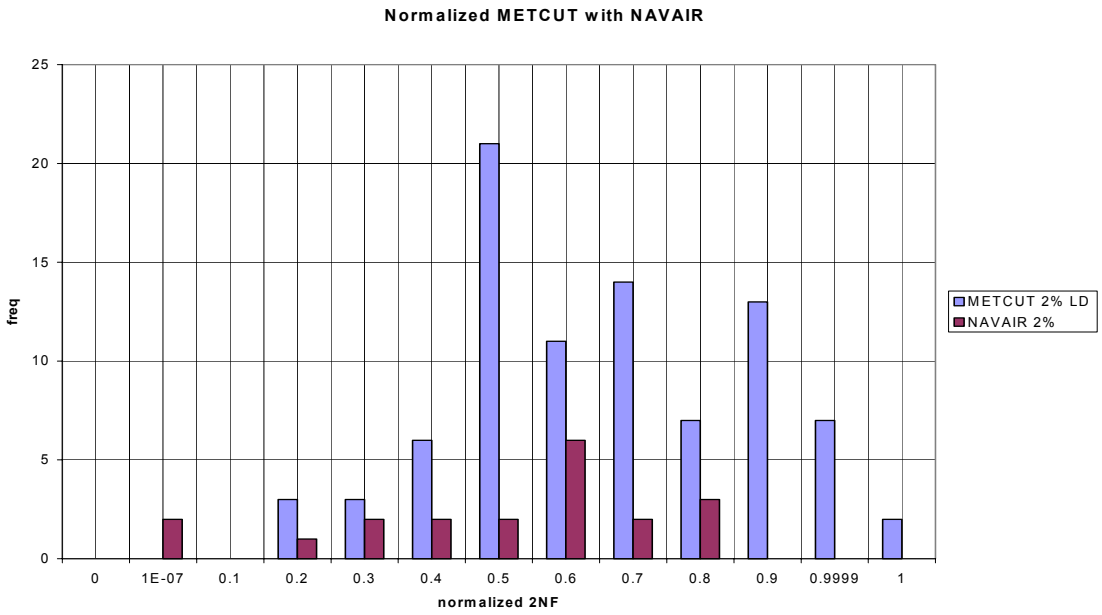

Figure. 10. Normalized Histogram

When all the data points are considered together in this normalized fashion, the data seems to portray a normal type distribution. Based on these findings, the investigator felt more confident assuming a normal distribution of data may ultimately

exist. However, making the correct correlation back into each range of 2Nf-strain level would be too hypothetical. It appears from figure (9), that the mean lies at about 50% of the maximum range and the standard deviation is at about 70% of the maximum range. When these correlations were later made in some Monte Carlo models, the results did not match the experimental data. That may be because of there was not enough data demonstrating the spread of the entire normalized sample set.

F. ATTEMPTS TO EXPLAIN THE DISTRIBUTION OF THE DATA AND THE BIGGER PROBLEM

With 15 data points at each strain level, it is puzzling that the data did not follow the standard normal distribution or a lognormal or Weibull distribution. It is often assumed that the fatigue data follows one of the standard distributions. However, a careful analysis must be completed of each material to determine the proper distribution before a probabilistic reliability method can be generated. Charles Annis of the American Society of Mechanical Engineers explains that "Computing a mean and standard deviation doesn't make a distribution normal (or even random). Actual behavior can be quite different from idealized behavior suggested by normal assumptions" [4]

The fact that the fatigue specimens were all cut from the same sheet of aluminum may be another clue. In order to illustrate this notion, the following experiment is presented. Let it be assumed that there are 3 things that could contribute to metal fatigue, micro cracks, grain alignment and test method. (Certainly there are a lot more factors than that, but this analysis is meant to present an idea.) Assume that the factors decrease the life accordingly. A 1 would mean that the material was essentially perfect in this category and that there was no loss of life due to this factor. A 5 would mean that the life would be limited by some factor, which would result in an average lifetime at a given strain. A 10 would mean that the factor was exceedingly bad and that the sample would have the shortest life. Using Crystal Ball® software, a Monte Carlo simulation was run.

(Monte Carlo simulation is the method in which random number generators are given parameters that match the distribution of the phenomenon of interest. The randomly picked random numbers are then mapped into the range specified by the distribution of the test sample set. Since the simulations are often run on computers, this

Monte Carlo simulation can predict the outcome of thousands of "events" or data points. This allows an investigator to make inferences about a very large population of samples vice the small set of test points that actually exist. This method is only as good as the degree to which the actual probability of the tests is predicted with accuracy.) [5].

One set of samples were established to have very high quality and very little variance. Random number generation picked numbers between 1 and 3 for each failure criteria. At each iteration, the result of the 3 numbers random generation was added to predicted the total fatigue damage inherent in that random sample. The simulation was run 1000 times, simulating 1000 random samples. Another sample set was made from weaker stock and had factors ranging from 1 to 9. Random number generation in this range would indicate that a few samples had very good characteristics, but there was also a wide range of samples, some of which have very poor characteristics (9s) . The following is a plot of the simulation. Since the Nearly perfect sample (blue) had 3 factors influencing fatigue, but those factors did not affect life significantly, there is a tight group of the probability distribution at about 7. In this simulation, that would mean that , on average, the blue samples had a summation of negative influences that would reduce life by a factor of 7. In contrast, the imperfect samples (red) had wide variance and a mean life reduction factor of 11.

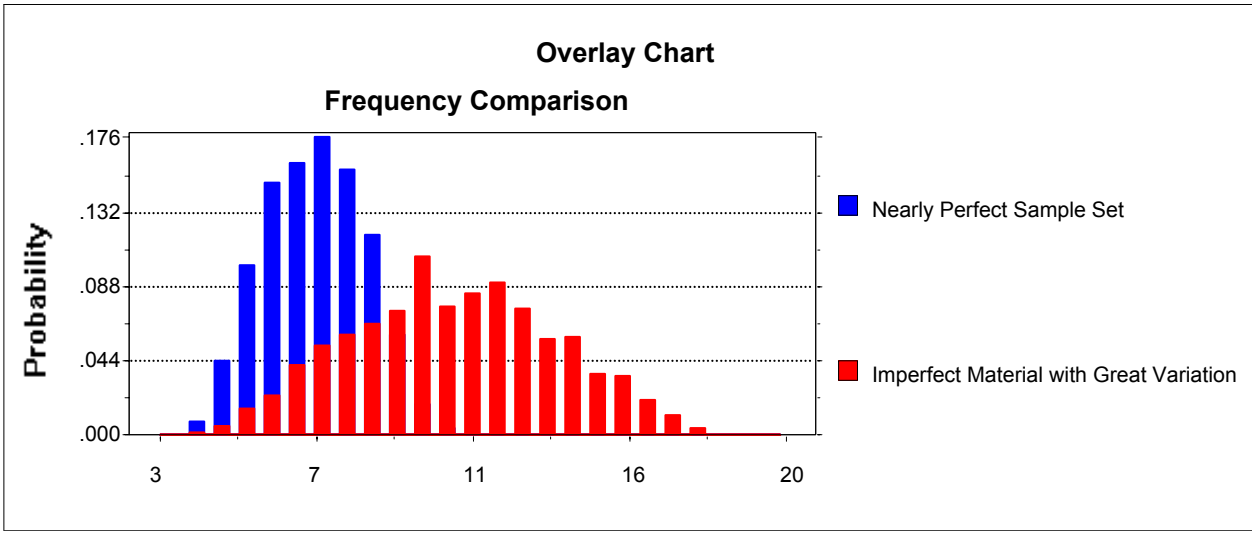

Figure. 11. Probability Distribution Explanations

It can be seen that independently, both simulations follow a normal distribution. However, when grouped together, they appear to follow a different distribution, possibly lognormal. Certainly, this simulation is a simplistic way to view the different distributions that often describe fatigue behavior. The purpose of this simulation was to propose the following question. When testing is completed, what do the results really show? Did the test group define the real characteristics of the entire population of the material or are "hidden" distributions resulting from material or testing variations skewing the results.

Another simulation was investigated which was more suggestive of the NAVAIR data. The Rusk and Hoffman paper, reference [3], describes the investigation of grain orientation in relation to the simulation. It was reported that there was a more favorable grain orientation. However, the grain orientation was not controlled during the initial experiments so that grain orientations were random during the testing process. The short transverse(S-T) specimens are modeled in a similar simulation, as was previously described, by the best life factors (1-4) and the long transverse specimens (L-T) are represented by the worst life factors (4-9). (An important note is that the grain orientation errors were not a result of unfavorable material characteristics, but were the result of testing errors generated from improper strain gage placement to the grains.) Never the less, the idea previously presented holds. If another simulation was run, comparing 1000 simulated best grain samples to 1000 simulated worst grain samples, two separate distribution would be produced. The following figure demonstrates this simulation.

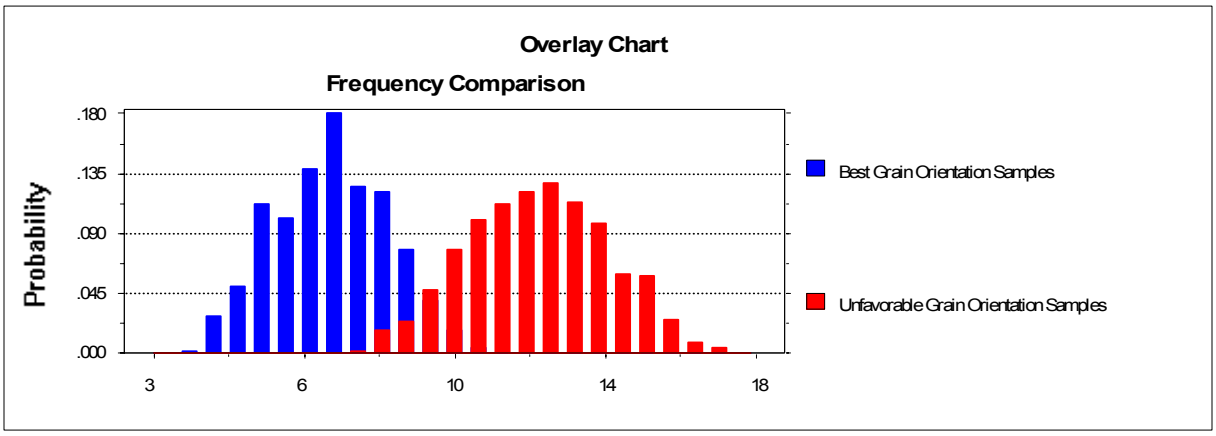

Figure. 12. Grain Orientation Simulation

Since the best grain orientation samples had better life reduction factors, the average reduction factor could be described as a 7. The worst grain orientation samples would be described as having a mean of 13. Both distributions demonstrate the normal trend. However, what would be the distribution of these same samples if they had not been sorted. The investigators would be left to figure out a way to fit some strange double-peaked distribution data. The investigator would be left wondering why his test data did behave according to the central limit theorem. Figure (12) demonstrates the same pattern that was found in Figure (11) with the actual test data. If the grain-controlling error had not been realized during testing, what type of distribution would have been defined for the material? What this simulation and investigation was meant to demonstrate was that the degree of material quality control significantly impacts the distribution of the sample. This, of course, comes as no surprise.

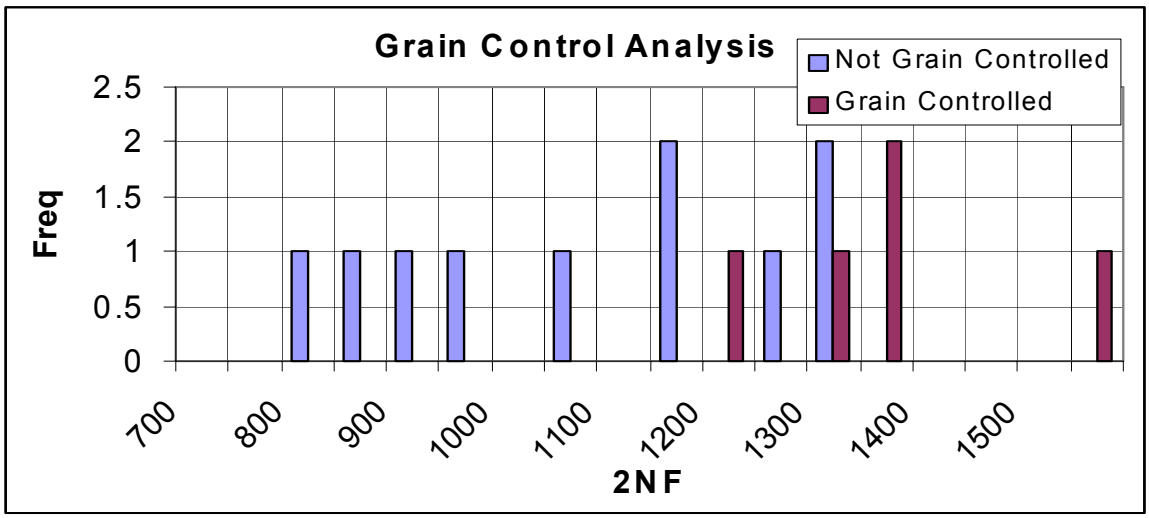

Figure. 13. Grain Control vs. Not Controlled Ref [3]

In an attempt to predict the fatigue-life characteristics of AL 7050 with the highest degree of accuracy, the NAVAIR test was very well devised to eliminate variability. There is another problem that presents itself. The tests were conducted by two, independent testing facilities, NAVAIR laboratories and METCUT laboratories. Even though the samples were all cut from the exact same sheet of metal with the same grain orientations, there still developed a large error between the data samples.

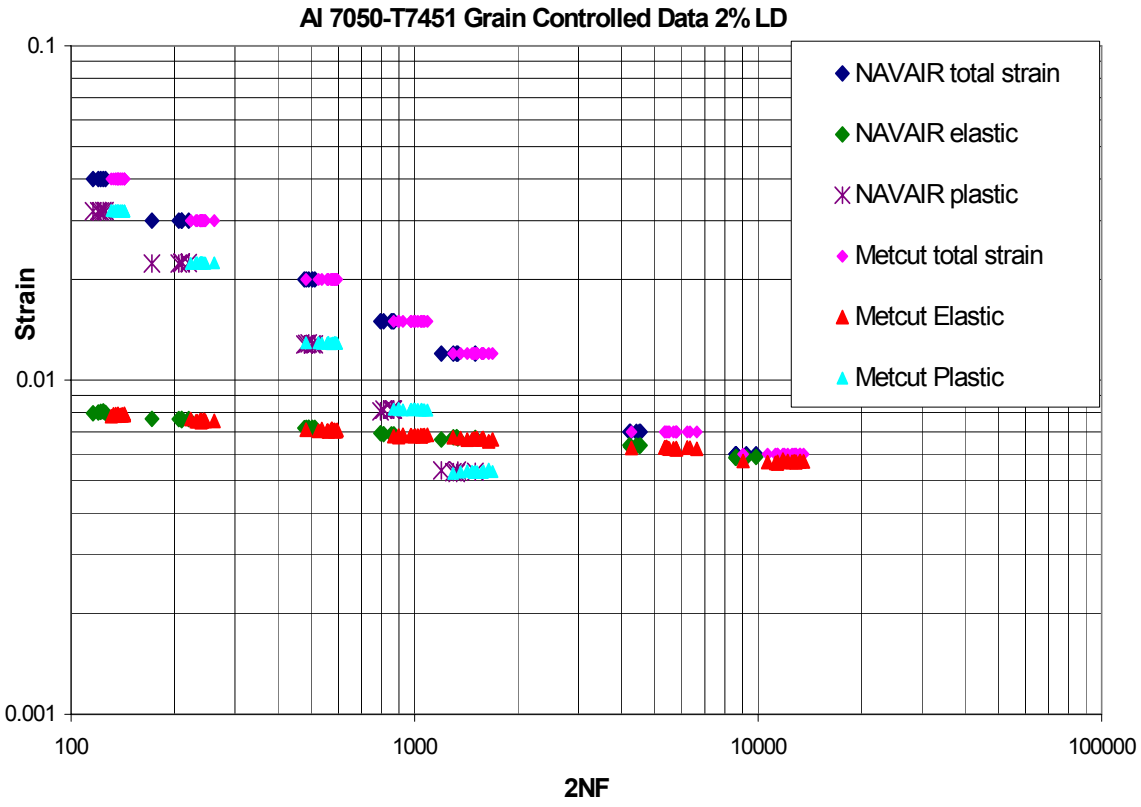

Figure. 14. Testing Laboratories Comparisons

It can be seen that the NAVAIR average life times are significantly shorter than the METCUT life times. This can be visualized with the following figures that demonstrate the probability characteristics of a combined set of data (NAVAIR and METCUT 2% load drop samples).

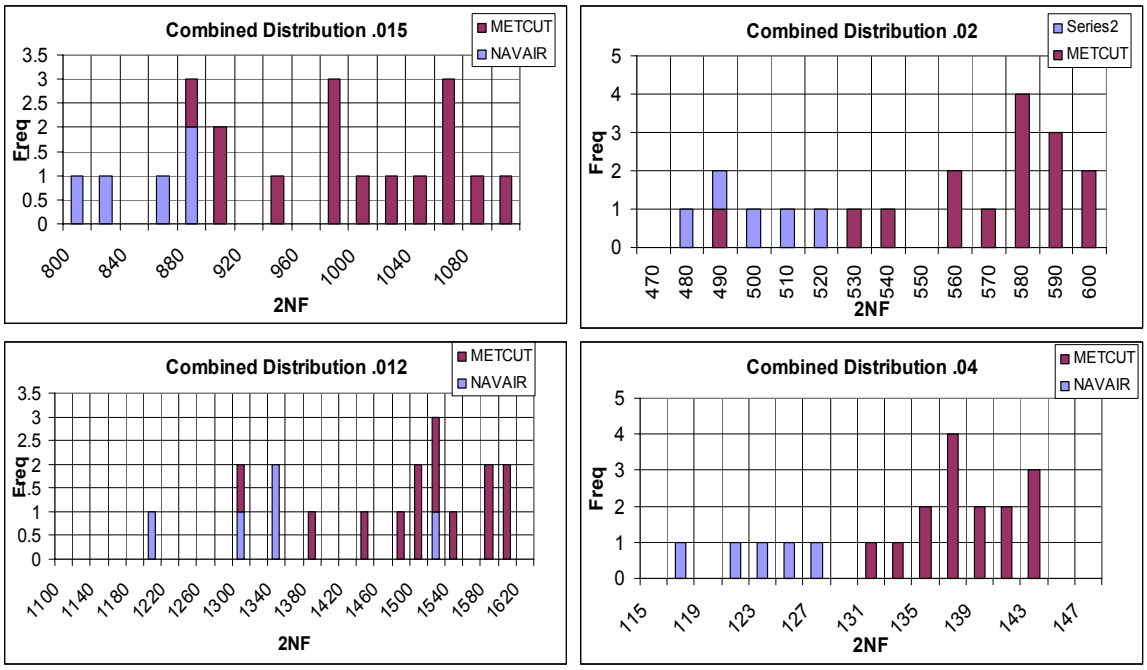

Figure. 15. Combined Distribution Histograms

Figure (15) depicts the weak normal distribution of the METCUT data combined with the few data points from the NAVAIR data, which essentially creates a "tail". The NAVAIR data obviously distorts the probability characteristics of the METCUT data. Does that mean that the NAVAIR data should be disregarded since there is only a few samples and they do not appear to fit in with the METCUT data? Suppose that an assumption is made that the NAVAIR data is erroneous, and it is not used in the probability-based model. The METCUT data is used exclusively, and later testing confirms a distinctive normal distribution about the current mean. With the distinctively defined METCUT data, probabilistic models can be developed that will accurately predict the strain life of the METCUT tests. That information is then transferred to the engineers assessing the life of the Navy/Marine Corps's aircraft. Since the distribution of the METCUT data is so well defined, the probability models infer clearly where the third standard deviation from mean lies with respect to a given strain level. Therefore a 100% FLE (Fleet Life Expended) would be re-defined and the engineers would be able to

predict with the highest statistical certainty the probability of a .01 inch crack if the aircraft was retired beyond 100% FLE.

NOW SUPPOSE THAT THE METCUT DATA WAS ERRONEOUS and the NAVAIR data was accurate. The probabilistic model based on MATCUT would not b e accurate and the Navy/ Marine Corps would put a few more in the water. This all seems logical but what does it mean to the development of a probabilistic model?

In order to safely predict the probability density function of the fatigue life of a material it is essential to consider all possible variables that will effect the nature of the fatigue life probability. With the present test, all samples were cut from the same piece of material while specially controlling the grain direction. The investigator does not intend to propose that sample testing should not be highly controlled. However, a few questions arise with the correlation of highly controlled test coupons to real world aircraft components. If component life is based on a direct correlation of the probabilistic model of the coupon samples, then the investigator has serious misgivings. The investigator believes that highly controlled experiments are a requirement, but that purposeful variation of material should be introduced into the investigative process. This would help to quantify the real world variations that occur between components due to different manufacturers, different material batches and different standards, just to name a few potential variables. To account for such problems and variations, engineers have traditionally relied on Safety Factors and Scatter Factors. Does that mean that engineers should throw in the towel and stick to their older simpler methods? Maybe in some cases, but the ability to develop better models is only a function of desire and investigation. After all, the only reason probabilities are used in engineering is to attempt to quantify what which cannot be explained deterministically. In order to develop a model accurate enough to extend the lives of aircraft beyond the original engineering best guesses, a very comprehensive investigation of metal fatigue will need to be completed, building on and going beyond what has been done in the past.

The first step would be to establish a very controlled probabilistic base line of the most ideal of all specimens. This is essentially what NAVAIR has already implemented. The second step would be to carefully evaluate those variables that could affect the life of the material, and would be evident in the population of real world components. Some

examples would be to determine the probability distribution from the same metal specimen, but measured at a different company. Another would be to test samples that all had the weakest grain direction. After all these different distributions have been determined, the investigators would be able to compare how different variable affected the life and make reasonable assumptions about the probability distribution of the entire population. This would give the engineer much more confidence in his material properties database. The development of the probabilistic model should very carefully control each of the different variables that could possibly effect the fatigue life of the metal parts used in Navy/Marine Corps aircraft and then quantify what each variable does to the probabilities of the whole sample space of all possibilities of aircraft parts.

As an example, let it be assumed that a particular type of aluminum was used in 90% of all helicopter drive trains and thus it was of interest to develop a better probabilistic model in order to extend the service life of these components. Samples should first be drawn in a highly controlled grain oriented manner. The probabilities of the distribution of these nearly perfect samples should be evaluated. Unless it is known that all drive trains are manufactured such that the component grains are oriented in the most favorable manner, there should be samples tested that purposefully have the poorest grain orientation. Then the two probabilities should be evaluated together. In that manner, the developing engineer would gain some understanding about the value of grain orientation. In the same manner more grain-controlled samples should be cut from another manufacture's product. Then these distributions should be compared against the former, also measuring the degree of error introduced by different companies. This type of testing should be conducted until all known life-changing possibilities have been examined and related. Here is a very simplistic example, obviously, reality is several orders of magnitude more complicated! The figure depicts the simulated distributions of 4 simulated sets of test data. Company A's specimens are depicted in shades of red, and Company B's in shades of green.

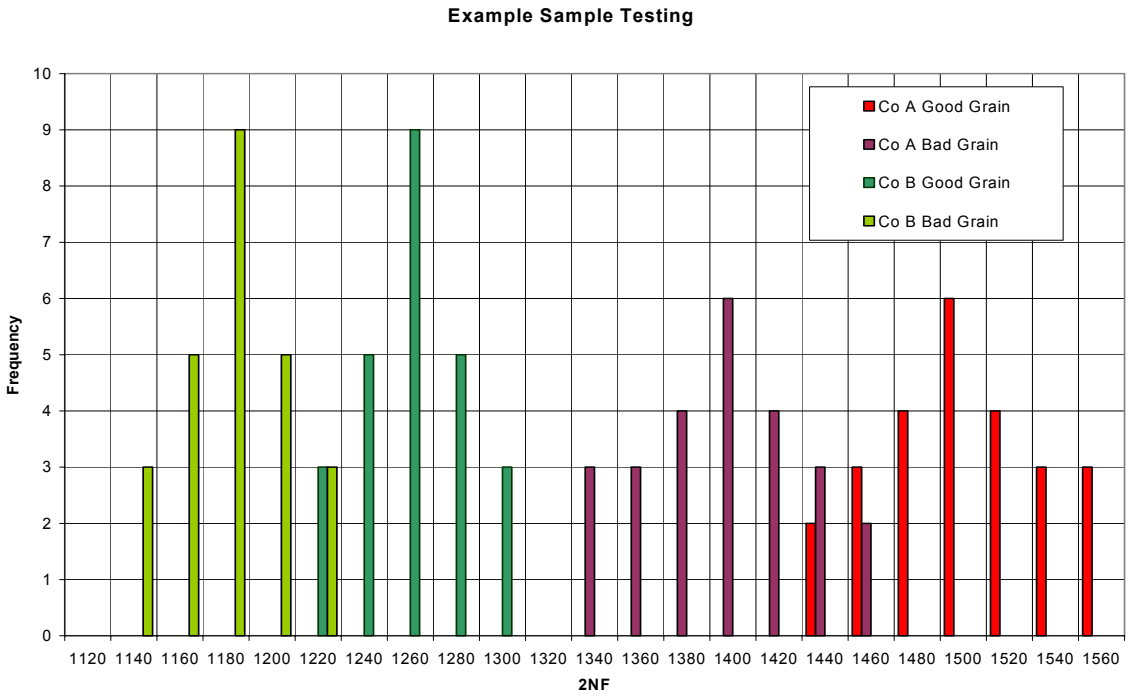

Figure. 16. Example Testing Distributions

What could does this tell the investigator? Company A makes a better product, though, with greater variance than company B. Poor grain orientation reduces life by about 7%, which is noted by the percent difference between the mean of good grains and bad grains of each material. If only these two companies make the parts in question and grain orientation is the only negative factor effecting fatigue life, then a fairly safe prediction could be made that the fatigue life of coupons (at this strain level) would be between about 1120 and 1580. The distribution may be difficult to characterize. A uniform distribution would be the safest bet in this case. If company C began to make the parts in question, this model would no longer hold and more testing would be required, since company C's parts have not been plotted and there is no way of knowing where they would fall. Unless "Management" is willing to pay for, at least this level of testing, no probability model will ever be able to safely provide a probability prediction beyond 100% FLE.

III. HYPOTHESIS

Since practicality dictates that only a small amount of testing can be conducted for a given material, the Monte Carlo simulation of the four strain-life constants $(\sigma_f', \varepsilon_f', b, c)$ should provide a matching data set of strain-life data points corresponding to hundreds of thousands tests. A measurement of the statistics of the simulation could be made and inferred to the actual material, thus completing the Probabilistic Strain-Life Model. The following is an illustration of Monte Carlo Simulation to model this material. (Note this is drawn from an early stage program "thesisdata4".)

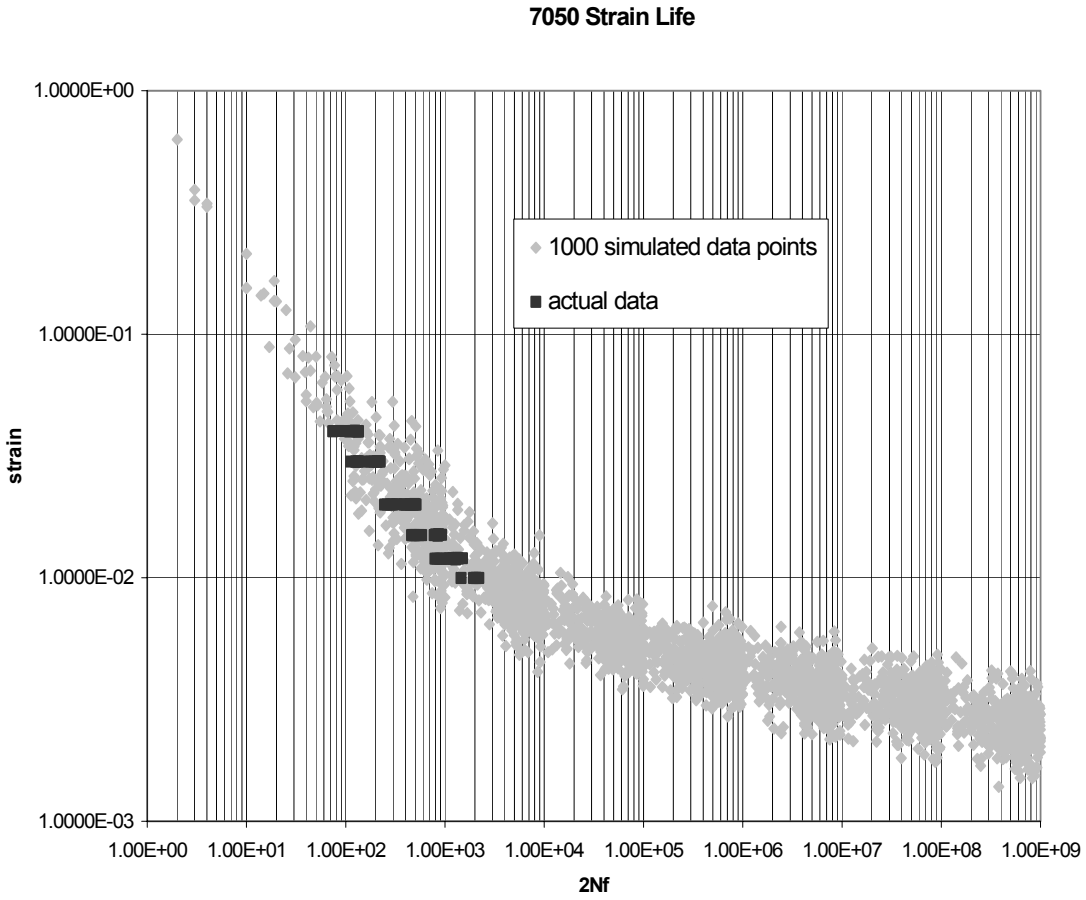

Figure. 17. Hypothetical Monte Carlo Simulation

A resulting probability distribution derived from the simulation is given in the following figure:

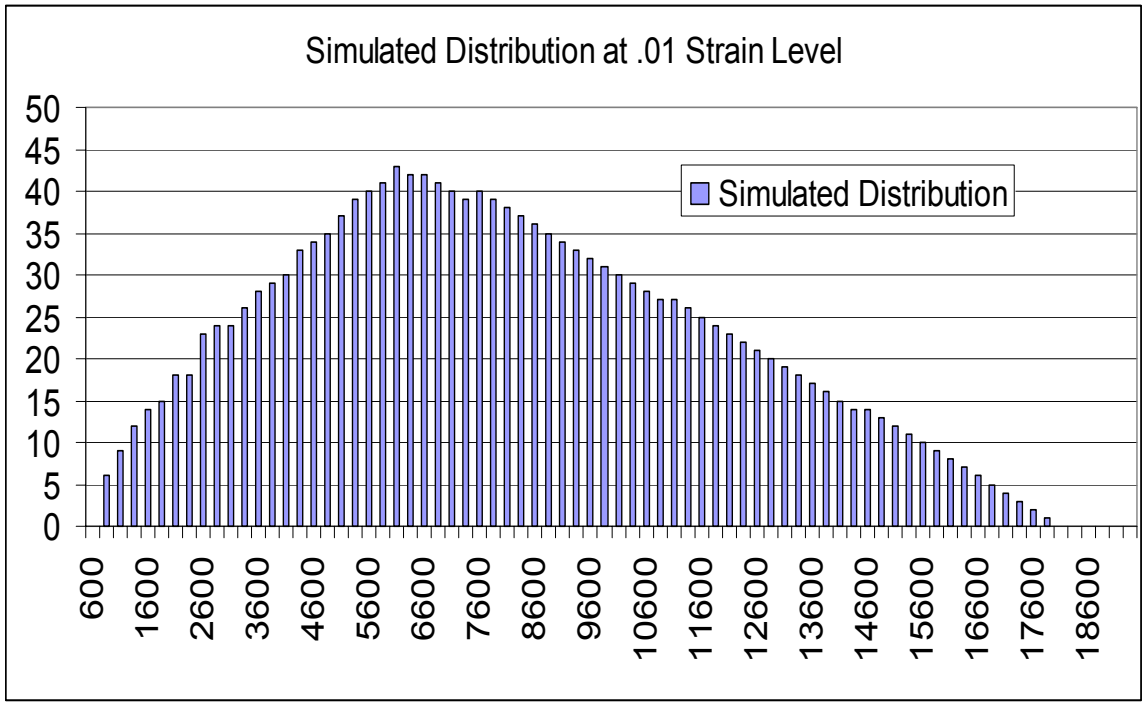

Figure. 18. Probability Profile from Monte Carlo Simulation

IV. SIMULATION METHODS

A. OVERVIEW

IF a distribution can be defined and related to the material's fatigue properties, then a Monte Carlo simulation model can be developed that will describe the solution to the variable strain-life problem at all levels, not just those defined by testing. Sufficient testing must be completed to correctly identify the distribution pattern of the material. Additionally, the test data must provide some sort of "anchor" on which to vary the model. This anchor could be the mean line of all data points or it could be the minimum or maximum. Somehow, a key parameter of the data must be defined to restrict the range of perturbation.

These models were developed with the assumption that variation would be about the Coffin-Manson strain-life equation, which is the best known fit to the mean line of the test data points. Additionally, a normal distribution was initially assumed to define the characteristics of variation. Many of the models were developed before the completion of all the sample testing. Therefore, the evaluator used the historical assumption of the strain-life equation and a normal distribution. After new test data was presented that obviously did not demonstrate a normal distribution, the investigator modified the simulations to reflect a uniform distribution. As has been previously stated, there could be problems with an assumption of the probability characteristics of a particular metal. However, the methods developed for simulation could provide insight and a baseline for the future development of strain-life models once the material is accurately measured.

B. MULTIPLE SOLUTION METHOD OF MONTE CARLO SIMULATION

The difficulty with the determination of life with the strain-life equation is that the best-fit function is highly non-linear and therefore requires a solver to determine the life (2Nf) given a particular strain of interest. In order to run a Monte Carlo simulation in this manner, the 4 strain-life parameters were randomly varied a determined number of times. At a given strain level, simulation created a large number of sets containing random combinations of the constants. These random sets were then used to solve the

strain-life equation for life(2Nf). The resulting output was a variation of life at a given strain level which was the result of randomly varying constants. The sequence was then repeated for each desired strain level. For this paper, algorithms will be used to more distinctively describe the method.

MULTIPLE STRAIN LEVEL SOLUTION MONTE CARLO ALGORITHM

1 . Establish number of simulations at each strain level (n).

2. Determine strain levels to evaluate.

3. Select the first strain level.

4. Generate (n) sets of 4 constants varied randomly (distribution dependant).

5. Use a numerical solver to solve the strain-life equation (n)-times for 2Nf.

6. Group the life values at that strain level.

7. Return to step 3 and compute new strain level until complete.

8. Determine the probability density function of life at each strain level

9. Determine probability parameters of interest (ex. Mean, 2sigma, 3sigma)

10. Connect the parameters of interest at each strain level

11. Plot

C. 2NF SEEDING SOLUTION METHOD OF MONTE CARLO

Since the previous method required the non-linear numerical solution of the life for each strain-constant set combination, computation time was significant. Large simulations often took several hours. For this reason, a simpler, quicker method was developed. Instead of solving the equation many times for life, a very large number of random, uniformly distributed, 2Nf data points were generated across the range of interest. Each 2Nf was matched with a sample set of the 4 randomly varied constants. Thus a strain level was directly computed from the equation. After all these strain levels were found, they were grouped into small ranges corresponding to a strain level. Once

that was complete, distributions were obtained as before. This method was significantly faster.

2NF SEEDING MONTE CARLO SOLUTION ALGORITHM

1 . Establish number of (n) random data points.

2. Generate (n) uniformly distributed random life points with in the desired range.

3. Generate the same number of random sample sets of the 4 constants for use with each life point.

4. Calculate the strain at each life data point.

5. Sort the data pairs into small strain intervals. (ie all between .015 and .025 would be called as .02 strain)

6. Evaluate the probability distributions of each strain interval

7. Determine probability parameters of interest (ex. Mean, 2sigma, 3 sigma)

8. Connect the parameters of interest at each strain level

9. Plot

THIS PAGE INTENTIONALLY LEFT BLANK

V. CORRELATION METHODS

A. OVERVIEW

After creating models that would compute Monte Carlo simulations of input parameters, there was a requirement to input the correct parameters that would model the characteristics of the material properly. During the model-building phase of this thesis, a normal distribution was assumed to model the fatigue life characteristics. Standard deviations of the life cycles at each level were established to be 10% of the mean at each level. Thus, the 4 strain-life constants (σ_f', ε_f',b,c) were set to vary about their mean with a 10% standard deviation. This approximate method provided the initial basis for the Monte Carlo simulation. As the process was refined, many more correlation methods were developed and will be discussed in later sections.

B. 2NF VARIATION EQUALS PARAMETER VARIATION CORRELATION

As has been described, the variation of the 4 strain-life constants with the same probability distribution type was the simplest method of correlation for a Monte Carlo simulation. Other, similar, probabilistic attempts validated the possibility of this method. The investigator found a paper prepared for the Virginia Transportation Research Council, [6] which computed the probabilistic fatigue life of bridges in a very similar manner. The bridge investigators modeled the damage stress function(h[B(ω)]) parameters (θ, m) with the same distribution parameters as the damage stress function. [Note: This paper was extremely important to the investigator of this thesis since his father drove over the bridge in question on his way to work.]

What type of distribution to use became difficult to determine since the actual experimental data did not demonstrate any clear profile. Since the test data did not demonstrate a solid normal distribution, the constants were varied with a uniform distribution. The range of the constants was of the same ratio as the actual test data mean to maximum range. The test data values and the strain-life constants were imported from the Excel® file into a MATLAB® file ("strainlife9.m" Appendix B). That program computed the mean value and range of each strain level. The average data variation was

then computed by dividing one half the range at each level by the average life at each level. These values were then averaged to determine the mean variation of the uniform distribution of each of the 4 strain-life constants during Monte Carlo simulation.

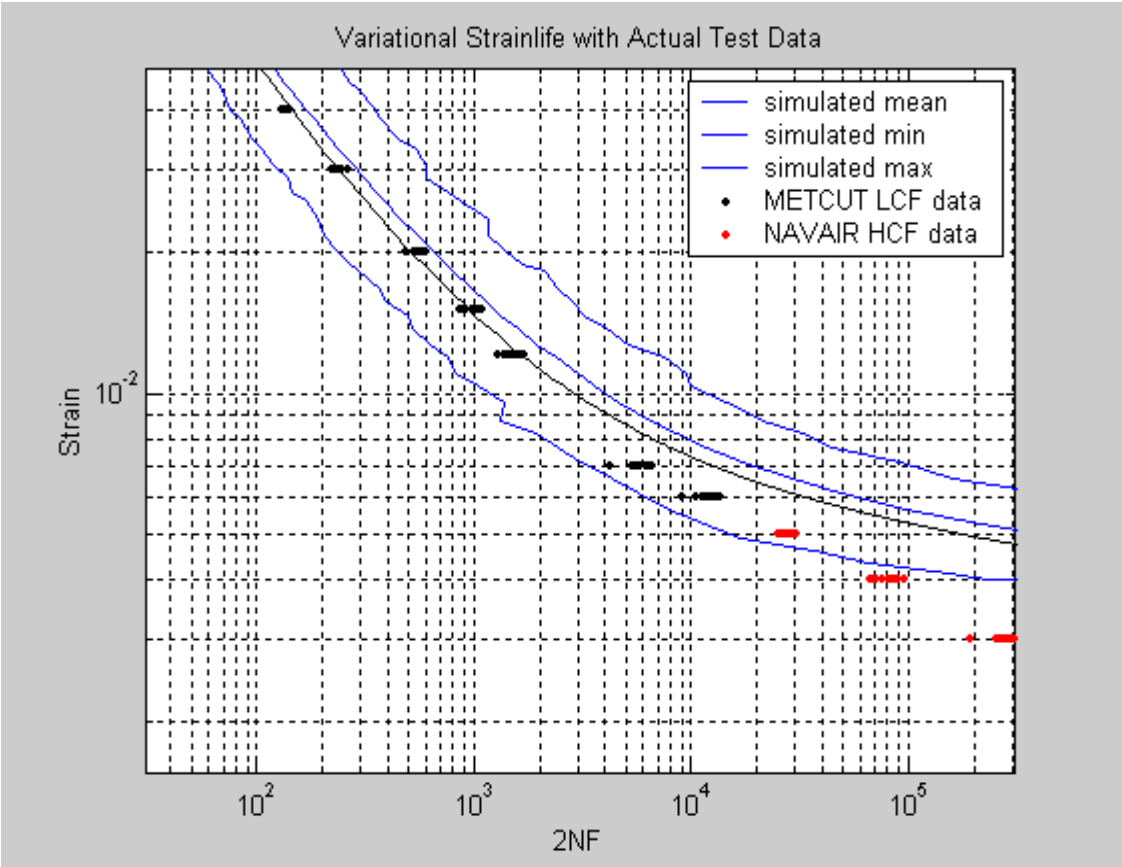

Figure. 19. Monte Carlo Simulation of Strain-Life Equation: Uniform Variation of Strain-Life Constants

The resulting probabilistic profile was entirely too wide. Variation of all 4 constants by an averaged amount of the 2Nf variation allowed for too much variability into the solution. The investigator experimented with varying the degree to which the 4 constants were varied. Each parameter had a specific and significant impact on the output. Large variations of the ductility constant resulted in very wide bands of scatter in the low cycle region. Conversely, large variations of the strength coefficient resulted in wide bands of scatter in the high cycle region. Variation of the exponents had similar yet magnified effects. Nice fits of the data could be obtained by visual adjustment, in other words, trial and error. Unfortunately, that method lacked a clear correlation to the

data and was simply an engineering adjustment . (Square peg into round hole with big hammer)

In order to visualize the impact the fatigue life exponents have on the Monte Carlo solution, a simulation was completed in which only the 2 exponents (b,c) were varied with the test data degree of variation. The other constants, σ_f' and ε_f' were held constant. The following figure displays that simulation.

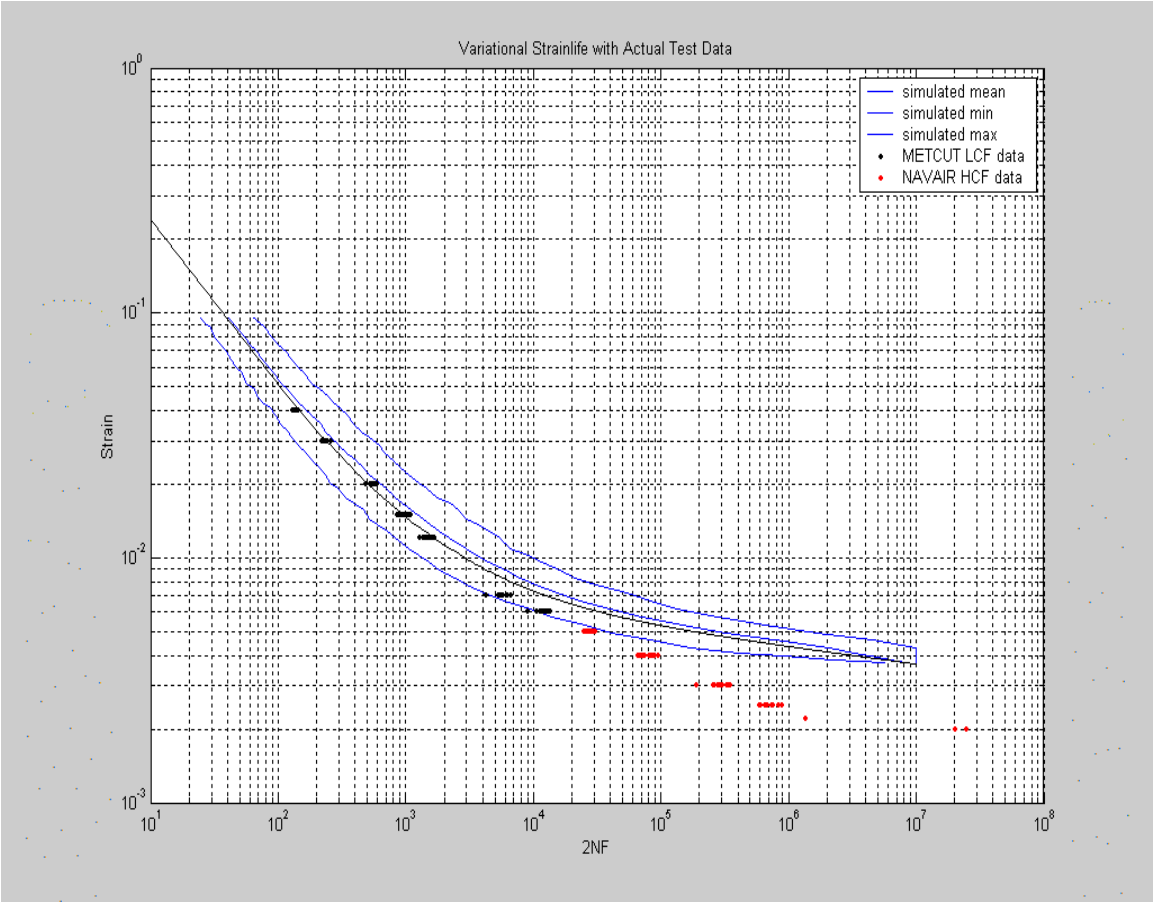

Figure. 20. Monte Carlo Simulation of Strain-Life: Variation of Exponents b and c, only

The exponents had a significant impact on the scatter of the Monte Carlo solution. The next chart will demonstrate that it was the variation of the exponents that contributed the most to the scatter of the simulation. Of course this only makes sense considering the mathematical significance of an exponent compared to a coefficient! If the exponents were held constant, the resulting simulation distribution of only σ_f' and ε_f', appears to be very good.

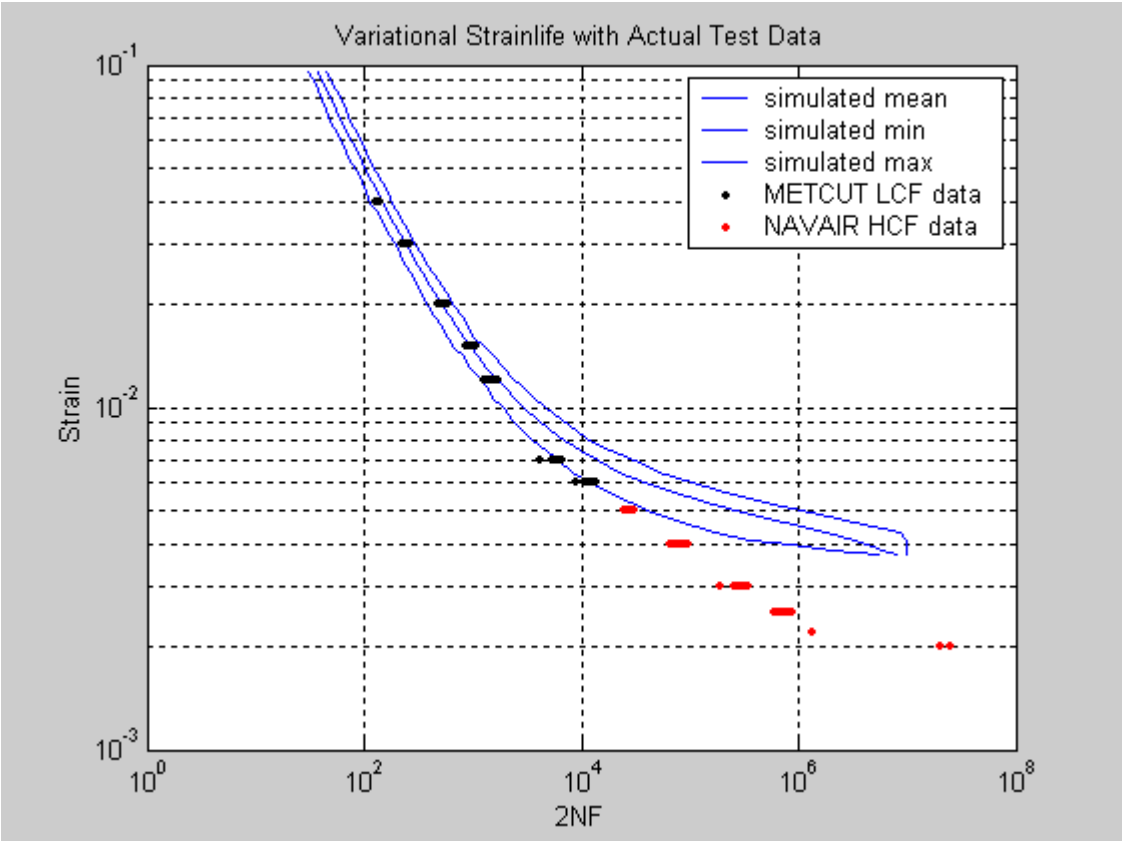

Figure. 21. Monte Carlo Simulation of Strain-Life: Variation of σ_f' ε_f' only

Variation of the 2 coefficients, σ_f' ε_f' , provided the best solution to the model that set parameter variation equal to the data variation . Although this method seemed to describe the nature of low cycle fatigue, the figure demonstrates some obvious deficiencies. Simulation with an averaged range doesn't accurately describe the data spread at every sample level. Possibly with a large sample of the population, these inconsistencies would disappear. The greatest problem is with the strain-life equation itself. It can readily be seen that the mean line rapidly diverges from the actual data in the intermediate range. Since the high cycle data was generated only from the NAVAIR laboratory, it is possible that high cycle fatigue samples tested in the METCUT lab may have been a little closer to the mean line.

As a comparison of Monte Carlo models, the solution method (solving for 2Nf, instead of seeding with 2Nf) was also computed using a 10% uniform parameter variation of the 2 coefficients. The results are almost identical except for their output

characteristics. Unfortunately, the solution method of Monte Carlo simulation is numerically intensive and took 30 minutes to generate. This is contrasted to the 2NF seeding method, which completed the task in about 3 minutes. This program was called "strainlife8a.m" Appendix C. Due to the time required to run this type of simulation, the 2Nf seeding method will be used exclusively through out the remainder of this report.

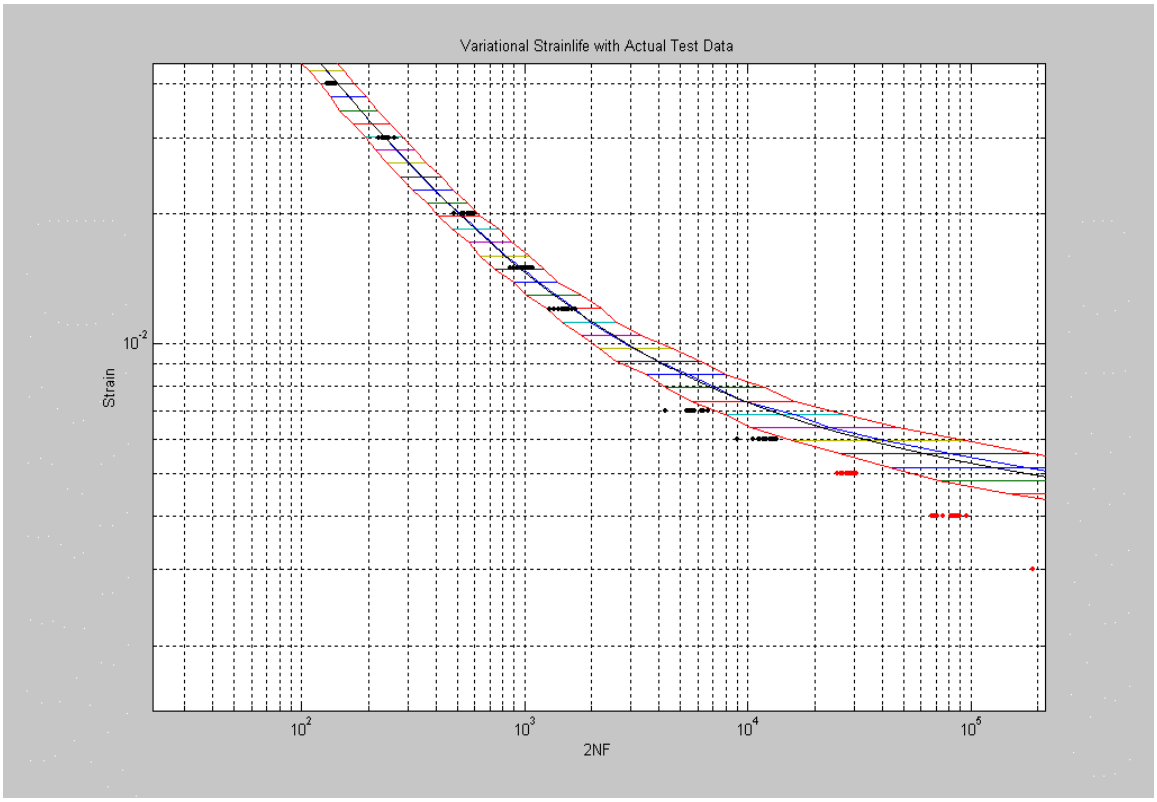

Figure. 22. Coefficient Variation by Solution Method of Monte Carlo Simulation

C. DATA PROJECTION/PARAMETER VARIATION METHOD OF CORRELATION

Another approach to predict life (2Nf) is presented in this section. Instead of computing the final solution's (2Nf) as a result of parameter variation during the Monte Carlo simulation, a more specific estimation was derived. The approach was to break the solution into the elastic and plastic parts of the equation and evaluate the contribution of each parameter to the entire solution. This method would allow different variations between parameters. Once the elastic and plastic strain regression lines were obtained, the data points were projected about the regression slope back to the Y axis. This method

assumed that the slopes (b,c) were constant. Using constant slopes the scatter about $\sigma_f^{'}$ and $\varepsilon_f^{'}$ was obtained with the following.

$$\varepsilon_{a,elastic} = Y_{\text{int}} \times (Nf)^b$$
$$Y_{\text{int}} = \varepsilon_a / (Nf)^b$$

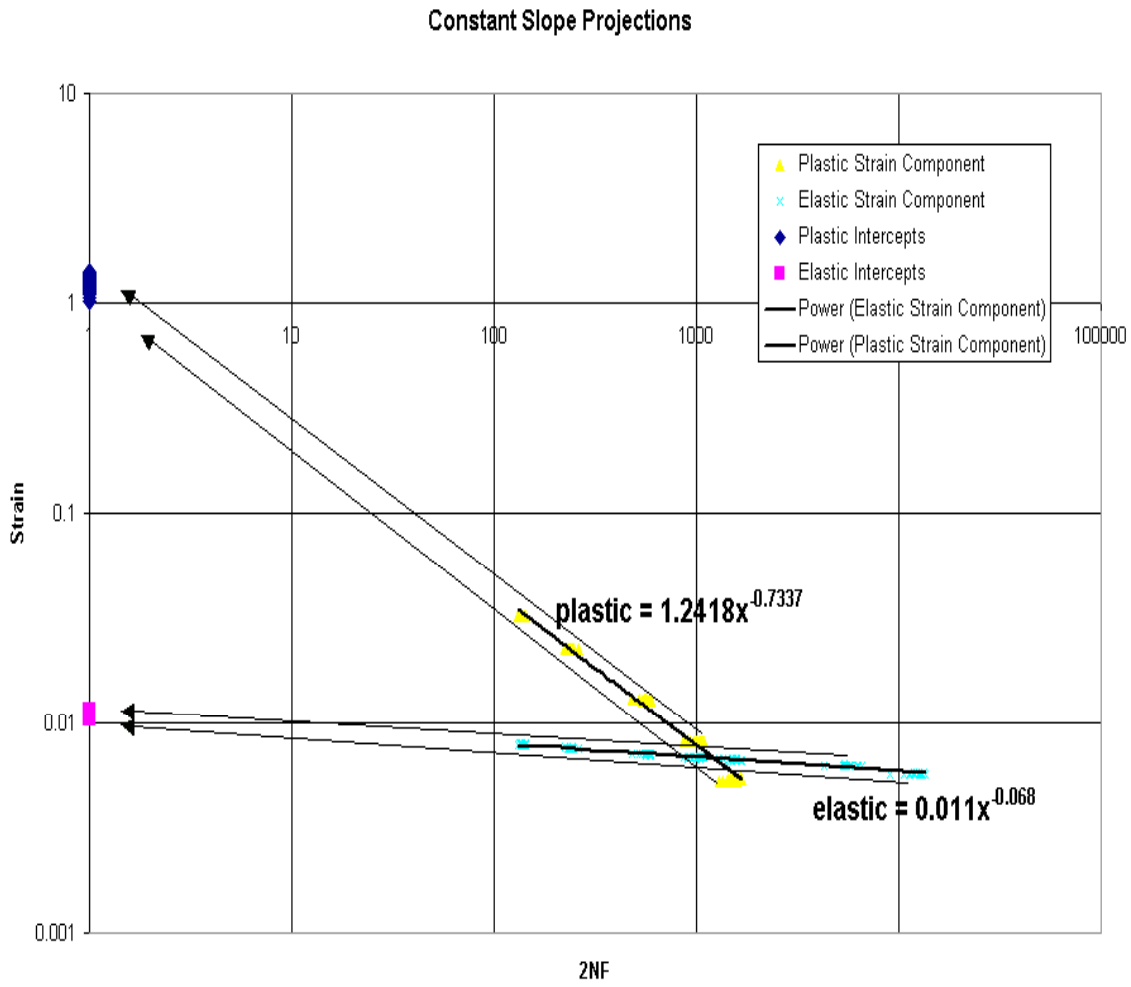

Figure. 23. Constant Slope Projection Visualization

This method demonstrated a greater variability for the fatigue ductility constant $\varepsilon_f^{'}$, than the fatigue strength constant $\sigma_f^{'}$. After projections the following data was obtained:

Table 2. Elastic and Plastic Y Intercepts

	Elastic Ints	Plastic Ints
max	0.01139211	1.39966605
min	0.01074658	1.0570559
range	0.00064553	0.34261015
mean	0.01096709	1.13431111
%dev	0.02943036	0.15102124

This approach quantified the larger variation of the data spread for the plastic points as compared to the elastic points. In the correlation model, discussed in the previous section, all parameters were varied with the same degree of variation as the final solution. Since it was shown that the variation of the exponents adversely affect Monte Carlo simulation, b and c were held constant. The fatigue ductility constant, ε_f' , was varied with a uniform distribution with a range from 1.06 to 1.4. The fatigue strength constant, σ_f', was varied with a uniform distribution of range .0107 to .0114. The following figure shows the resulting Monte Carlo simulation.

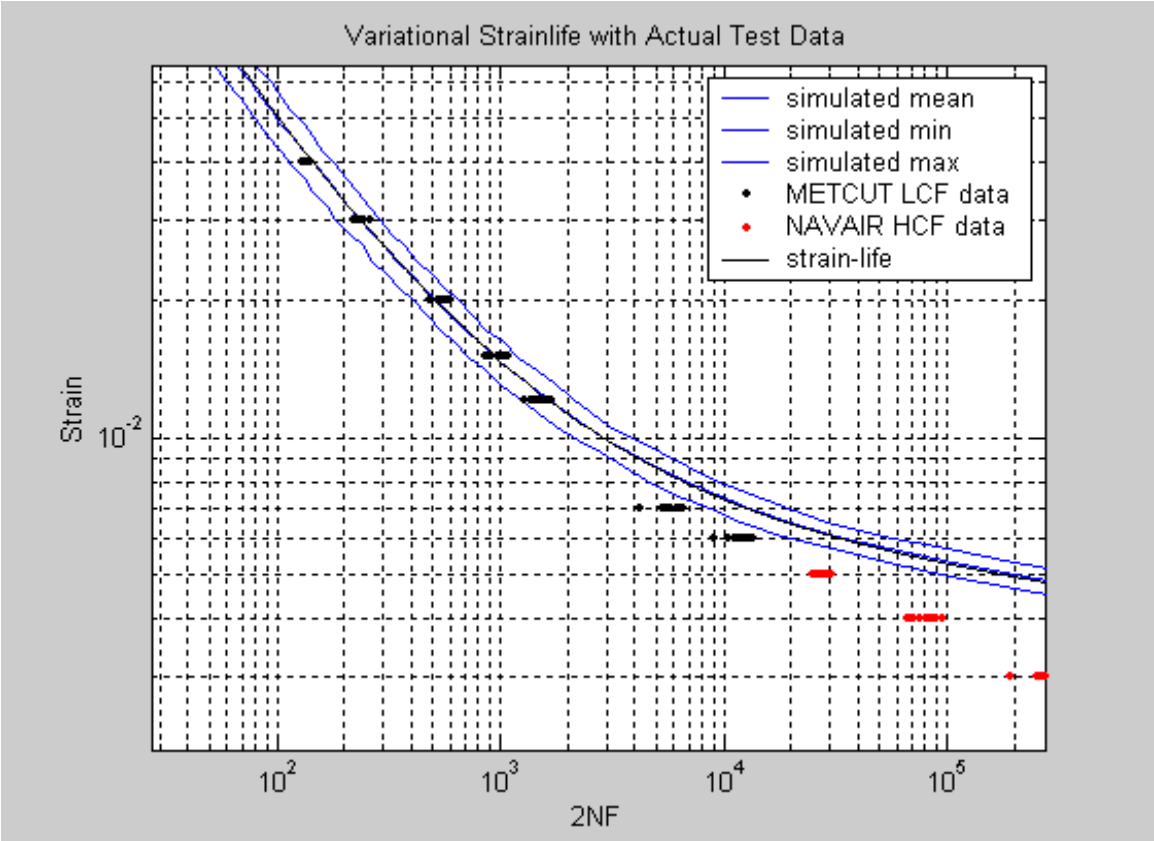

Figure. 24. Slope Intercept Correlation Method

This method demonstrated good correlation through the LCF range, although the solution was erroneous for the HCF range for reasons previously discussed. This model has a subtle difference to the previous two, coefficient variation models. Although they both appear almost identical, the second method results in a smaller scatter band in the HCF range. In the "solution variation equals parameter variation" model there was a wider band of variation in the HCF range. This is because the fatigue strength constant was only uniformly varied by about 3% from the mean in this model instead of 10% in the previous model.

VI. BEST SOLUTION METHODS

A. OVERVIEW

It became apparent during the course of this study that there was no simple solution or Monte Carlo simulation that would correctly model the present test data and provide a reasonable probabilistic model. This was due in part to the fact that there was not enough data collected at the strain levels to clearly define a distribution. Additionally, the fact that two separate laboratories' test results were used, one lab providing LCF and one lab providing HCF, caused the test data to be ill-behaved to the standard strain-life equation model. The investigator realized without an accurate prediction of some parameter of the test data (probably a mean value), that accurate probabilistic model could never be built. For this reason, the investigator attempted many methods to establish some function that would accurately predict all mean values. Such methods included developing an algorithm that would fit a line between each mean data point, the use of alternative functions to the strain-life equation, incorporation of an 8 constant strain-life equation, and the use of evolutionary algorithms and genetic algorithms to model the test data. The final result should come as no surprise. There are an infinite number of ways to fit the data points, each with it's advantages and disadvantages, accuracies and inaccuracies. The best-fit model developed for this material would in no way have any correlation to any other material. However, the methods and concepts evaluated herein may be of adopted for other material behavior. However, each new material will require a rigorous investigation to create a best-fit probabilistic model for each particular material. As in all engineering, there are no easy solutions! The remainder of this chapter will discuss the development of some of the more useful and accurate methods.

B. STRAIGHT LINES METHOD

The most obvious and simplest method that could be useful in the development of a probabilistic model is the use of fitting a straight line between the mean data points. This model would work very well for tests in which data for many strain levels were

available. In the limit, of an infinite number of strain levels tested, there would be a smooth curve. Upon completion of the accurate determination of the mean points, and with a solid understanding of the actual distribution of the fatigue scatter, Monte Carlo simulation could be run to define the scatter distribution outside the tested range. An even more simple method would be to use the distribution profile described by the data at each strain level and connect alike probability likelihood estimators. The following figure shows such a curve and is obtained from "strainlifelinz.m" Appendix D.

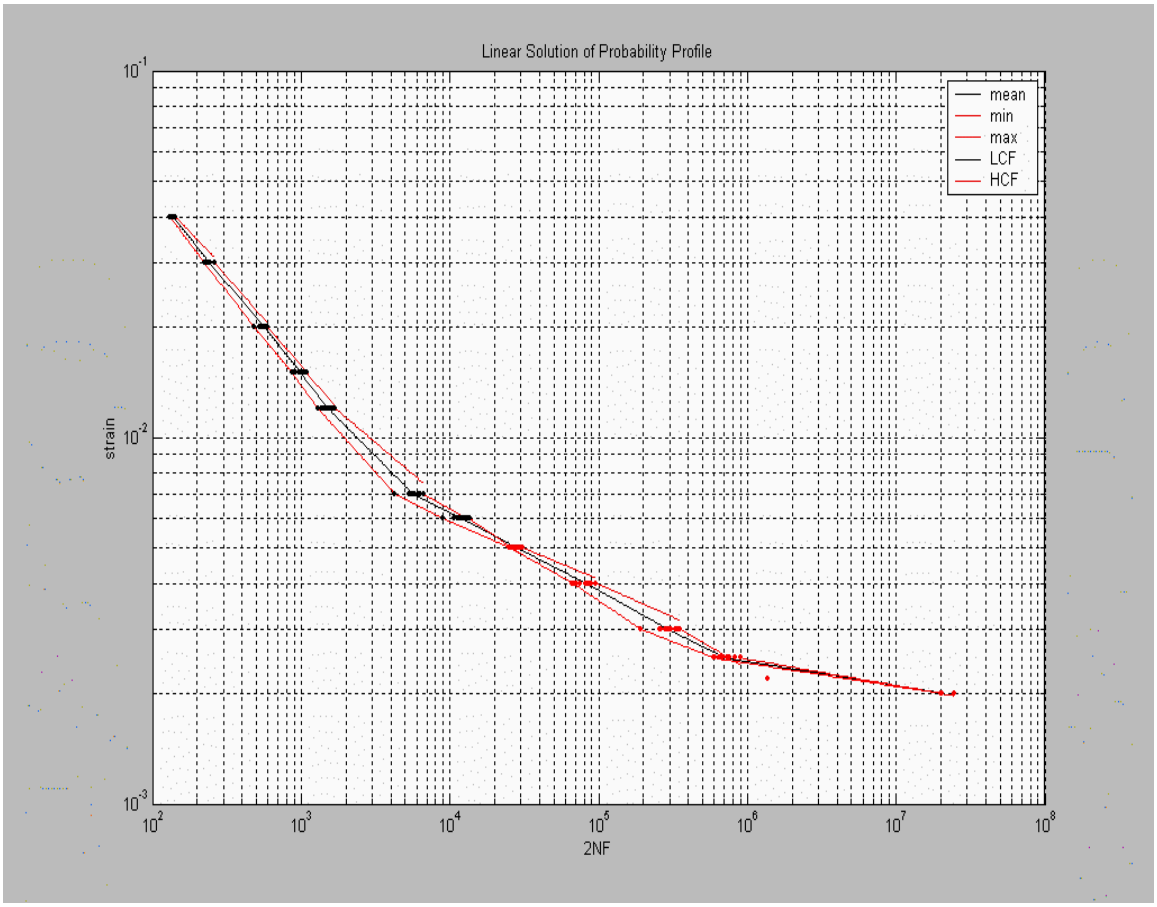

Figure. 25. Linear Solution of Probability Based Fatigue Life Model

This depiction does not look like a good solution, however, it is the most accurate of all solutions based on the data obtained from testing. The red bands depict the maximum and minimum values that were obtained from testing. Since there was no conclusive probability distribution obtained from the data, the best guess for a probability distribution would be a uniform distribution between the maximum and minimum points.

If a distinctive probability distribution could be obtained from the data, then the defining parameters could be incorporated to enhance this simple solution.

C. 4 PARAMETER PIECE-WISE ALGORITHM

Having previously established that the 4- parameter, Strain-Life equation could not fit all the test data points effectively, the investigator improved the previous straight-line method. This method involved breaking the data sets up into pairs as was done in the previous straight-line method. Instead of fitting a straight line between the points, a curve defined by some variation of the Strain-Life equation, was fit between the two points. Essentially, each segment was created from the Strain-Life equation with it's own particular set of parameters.

PIECE-WISE, 4 PARAMETER STRAIN-LIFE ALGORITHM

1 . Create (uniform or normal) random vectors of the 4 parameter possibilities for each parameter (σ_f' , ε_f' ,b,c) .

2. Evaluate the 4 parameter (σ_f' , ε_f' ,b,c) equation at the first data point.

3. Find all the combination sets of parameters that provide the solutions within a specified tolerance from the known data point.

4. From those sets of best fit parameters, find the set that was a best fit to the next data point.

5. Plot the function from the first data point to the second data point.

6. Establish the second data point as the new first data point and then repeat Steps 1 through 5.

The program "strainlifega.m", (Appendix E.), was developed to evaluated the best fit to the data and to run Monte Carlo 2Nf seed simulation at the same time. The following figure was produced from a run of this program.

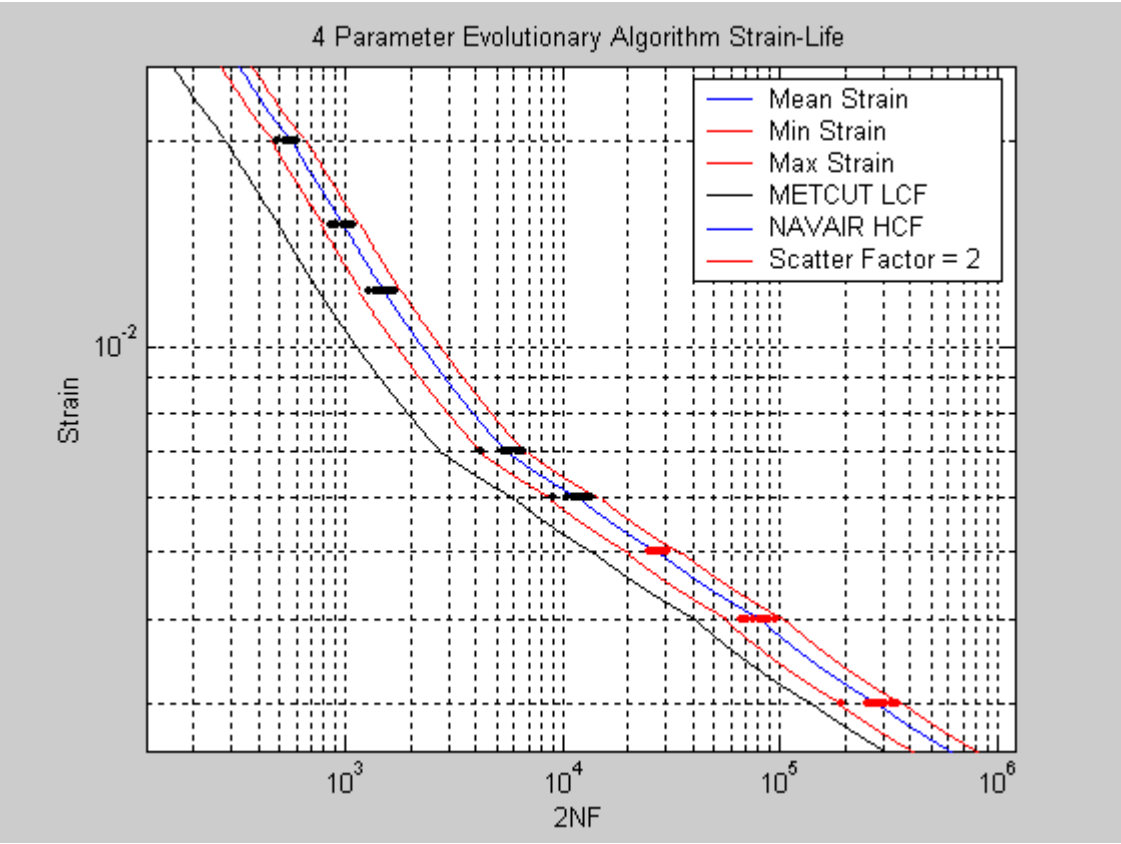

Figure. 26. 4 Parameter Evolutionary Algorithm

Because each partial solution was determined between two data points, the final solution is very similar to the straight-line method. This method does allow for some small amount of curvature to be introduced between points. This further optimizes the probabilistic model. This model only defines an average range of variation based on the actual data. If more conclusive test data was available, the probabilistic model could also include probability bands. This chart further demonstrates the possibilities that a probabilistic model might have over the conventional methods. In chapter 2, NAVAIR's definition of FLE was defined as ½ the mean life. **(<u>If !)</u>** further research positively identifies the range of scatter to be that described by this model, then there would be a distinct advantage in using the probabilistic model to extend service lives beyond 100% FLE.

D. POSSIBLE 8 PARAMETER SOLUTION

After attempting to make the strain-life equation fit the test data with a variety of different techniques, the investigator realized that the irregularity of the data would not

permit a solution with only 4 constants. Experimenting with possibilities, the investigator rationalized that 8 constants had to be better than 4. Similar to a higher order polynomial in normal linear space, the investigator attempted to fit the data with the following equation.

$$\varepsilon_a = A \times 2Nf^b + C \times 2Nf^d + E \times 2Nf^f + G \times 2Nf^h$$

The following figure shows the results, demonstrating that 8 constants provide a significant amount of flexibility in crafting the fit of the data points and may provide for an exact fit to the test data.

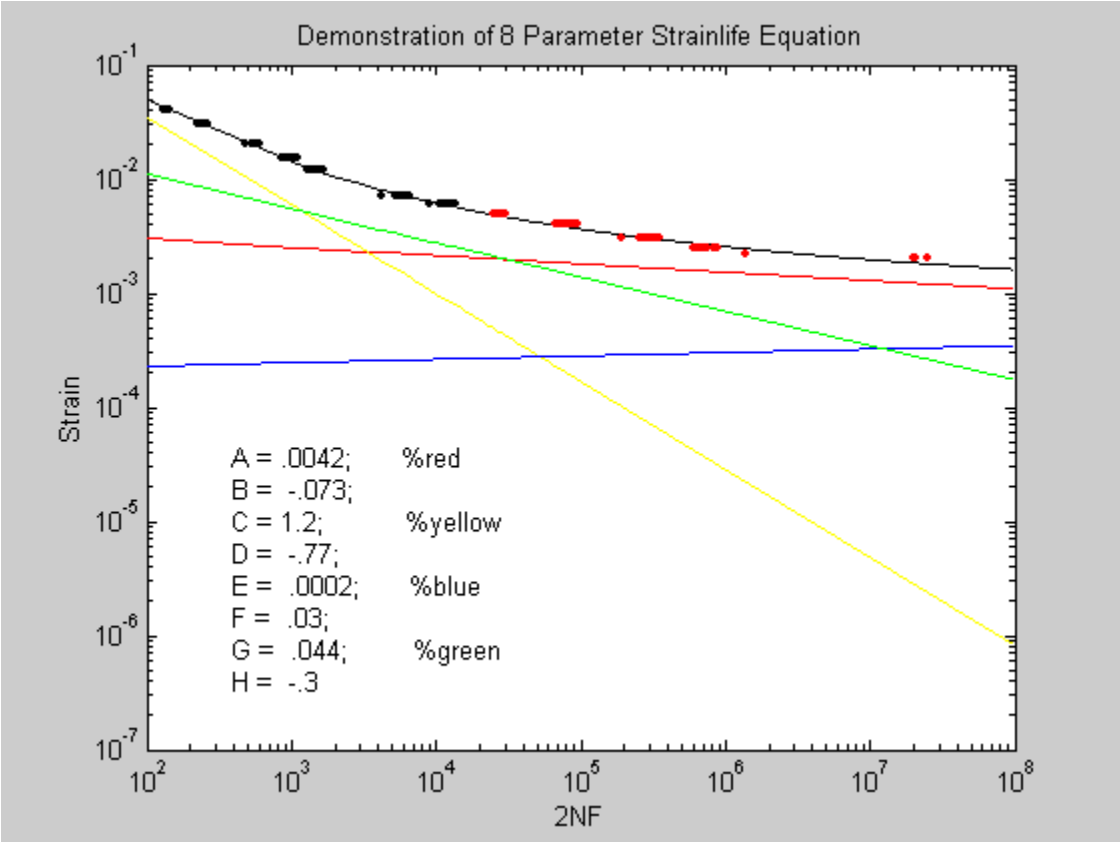

Figure. 27. Demonstration of 8 Parameter Strain-Life Equation

This fit, which was done in about 10 minutes by trial and error, comes very close to accurately predicting a mean line. The investigator believed that advanced methods of numerical analysis might find the exact solution. The use of 8 parameters may not always be useful in correcting the strain-life equation. This data set was particularly hard to fit to known functions. Other sample data sets may find the use of a six constant

equation and the present algorithm useful in determining a best fit to sample data. The coefficients and exponents of this 6 or 8 constant equation do not need to correlate to any physical parameter like σ_f' etc., however, the use of the known elastic and plastic parameters may provide an initial guess for the solution.

E. EVOLUTIONARY ALGORITHM (PSEUDO-GENETIC ALGORITHM)

Having determined that a better solution to *this* set of data points might best be fit with an 8 constant equation, the investigator tried a variety of methods to solve the problem. The investigator drew upon the idea of "genetic algorithms", which will be covered in more detail later in the paper. Essentially, the idea involved finding the solution, by attempting many different possibilities using random number generation. This algorithm was slightly more complex because random numbers were used to generate the exponential values, and then a non-linear, least squares curve-fit was used to evaluate the coefficients. After a large number of "tries" the best solution was picked based on the error of each possible solution.

RANDOM EXPONENT/ NONLINEAR LEAST SQUARES CURVE-FIT ALGORITHM

1 . Randomly pick 4 exponent values from within a prescribed range.

2. Pass the exponential parameters into a function that will determine the coefficients of that particular set based on a non-linear least squares curve fit.

3. Evaluate the weighted error of that particular set of 4 randomly varied exponents and the solution of 4 coefficients.

4. After a prescribed number of iterations, sort the sets for the least error.

5. Plot the function from the least error set.

This algorithm was used to generate "strainfit.m" Appendix F. The results are presented in the following figure.

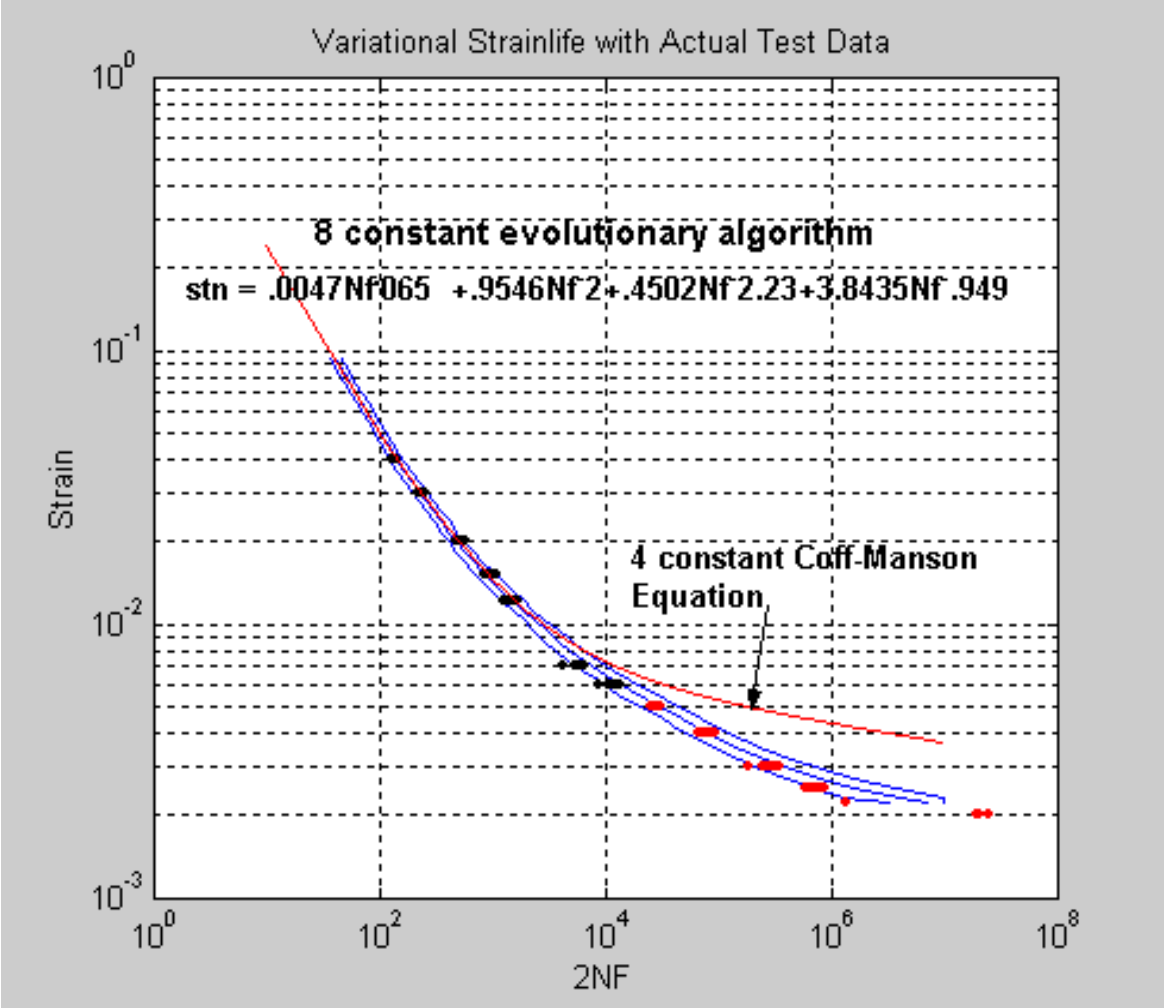

Figure. 28. 8 Parameter Curve Fit Compared to Coffin-Manson

This was the result of 5000 sets of random exponents with the matching computed coefficients. This method is an obvious improvement from the Coffin-Mason Strain-Life equation because it does a better job of describing the overall curvature of the fatigue life phenomenon. However, even with advanced solution methods, there was still a discontinuity of the solution at the knee. Possibly a better solution was found by the investigator's trial and error method!

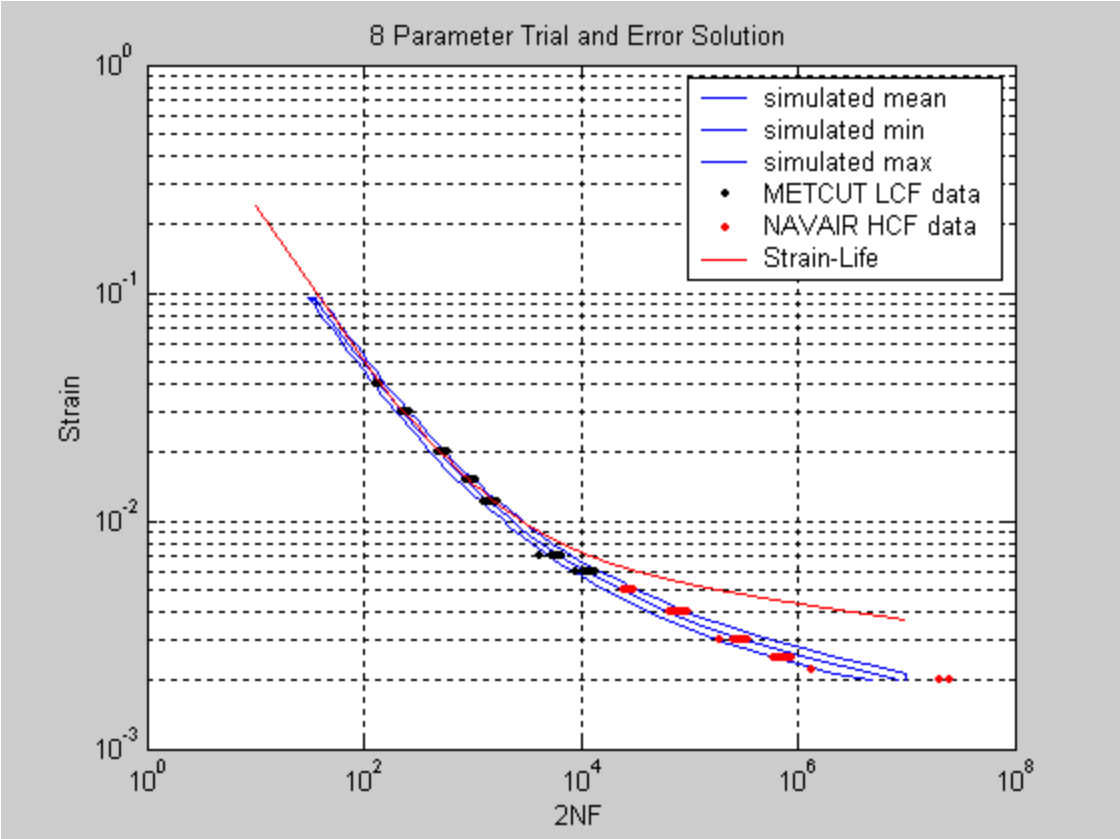

Figure. 29. Good Engineering Guess Rivals Complex Mathematics

Not that the investigator renounces the use of computers, but there is an irony that a solution which was completed quite simply in about 10 minutes is better than what a fairly complicated algorithm determined after 5000 iterations. Note that the trial and error solution only failed to accurately place one data point within the predicted range, while the computer solution failed to correlate 4 data points. This discrepancy was completely a function of the calculated mean line since both Monte Carlo simulations varied the solution about the determined mean line by exactly the same amount.

F. GENETIC ALGORITHM METHOD

Another attempt made by the investigator to lay the foundation for a probabilistic model, was with the use of "Genetic Algorithms". The idea behind such a method was to assume that a so-called "chromosome" could be encoded with information about the solution. The chromosome must be relatable to some parameter of the function. The chromosome is usually a string of '1's and '0's, 30 bits long. Thus a binary representation of any number between 0 and 1073741823 is possible. These numbers are

then mapped into the range that the variable is expected to lie within. Multiple chromosomes are then created (possible solutions). These possibilities are evaluated for their "fitness" to the desired solution. If none of the solutions are satisfactory, then a pseudo-random selection of the best chromosomes, subsequent mating, cross-over and mutation of the genes, results in a new population of chromosomes (answers). Eventually, after several generations, an answer is usually found. The difficulties with this method can be in relating a chromosome to some aspect of a fitness function of the problem. Usually, genetic algorithms are used to either minimize or maximize a solution. The selection of a fitness function to minimize or maximize in the case of this data set was particularly challenging. [7,8]

In an attempt to solve the 8-parameter Strain-Life equation, the investigator attempted to use genetic algorithms. Each of the 8 parameters was drawn from a 30 bit element of a 240 bit long chromosome. In this manner each particular chromosome represented a set of 8 parameters with a corresponding strain vector. The algorithm is provided below. [9]

8 PARAMETER GENETIC ALGORITHM

1 . Establish the length of the chromosomes and the population size

(# of chromosomes per generation)

2. Determine the probability of cross-over between chromosome "parents"

3. Initialize the chromosomes by setting each bit to a 1 or a 0 with the use of random number generation.

4. Import each chromosome individually into the function.

5. The function breaks the chromosomes up into smaller pieces representing the 8 parameters. (Formally, these steps are referred to as; concatenation, multi-parameter-mapping, and fixed-point coding.)

a. The binary information is converted to decimal and then mapped to a decimal value within the range specified.

b. After each part of the chromosome has been mapped into the appropriate range, the 8 parameter strain life equation is evaluated at the known 2NF mean values from each test range.

c. The strain vector answer is then compared to the actual test data.

d. The weighted, maximum percent errors are normalized which represents the magnitude of the error. This becomes the fitness function corresponding to the particular chromosome and is passed back to the main program.

6. After each chromosome from the population has been evaluated, the set is sorted by smallest fitness (error) to largest error.

7. The two best chromosomes (smallest error) are taken to be the king and queen. The king and queen chromosomes are the first two chromosomes for the next generation to ensure the best answers are not discarded randomly.

8. The king and queen are also mated by passing them to the cross-over program which determines for each of the 8 partial strings within the chromosomes, whether or not,(and if so, how much) there will be a flip-flop from the chromosome of one partial string to the corresponding chromosome of the other partial string. After each partial string has been evaluated and cross-over has occurred, the program determines if a small amount of mutation should occur and if so to which Allele. (Allele are the individual 1s and 0s). Allele mutation permits the possibility of a lucky solution being interjected along the way to a final solution.

9. The king and queen are mated approximately 10 times to produce 10 new chromosomes for the new population.

10. After the king and queen have finished mating, a larger set of new chromosomes is developed in the same manner. However, the parents are randomly picked from the old population for each set of new chromosomes. This set is called the “Clampetts”.

11. After all the Clampetts have mated, another completely random set is re-initialized, referred to as Melting-Pot. From this completely random set, approximately 20 more parent sets are selected and mated. This process allows for the rapid incorporation of new blood into each generation and prevents the solution from stalling.

12. After a predetermined number of generations, the chromosome corresponding to the best answer for the present generation is correlated to the 8 parameters and the modified strain-life equation is plotted.

The real difficulties with this method involved, selecting a performance criteria. The magnitude of the positional errors alone was not enough to obtain a good solution. When only the difference between the test data and the solution was used, the genetic algorithm tended to fit a straight line through the data, exactly what the investigator did not want to happen. A method of evaluating the slopes between data pairs was also investigated. This method worked well to mirror the shape of the curve, but the curve did not lie on the line. Thus a combination of the two methods was selected. In addition to both slope and position errors, elements that were deemed the most significant were also weighted to draw the solution through those points or draw the solution toward a slope. The following figure is a plot from the MATLAB® file "8paramGA". This algorithm requires 19 separate MATLAB® programs. These programs are reproduced in Appendix G.

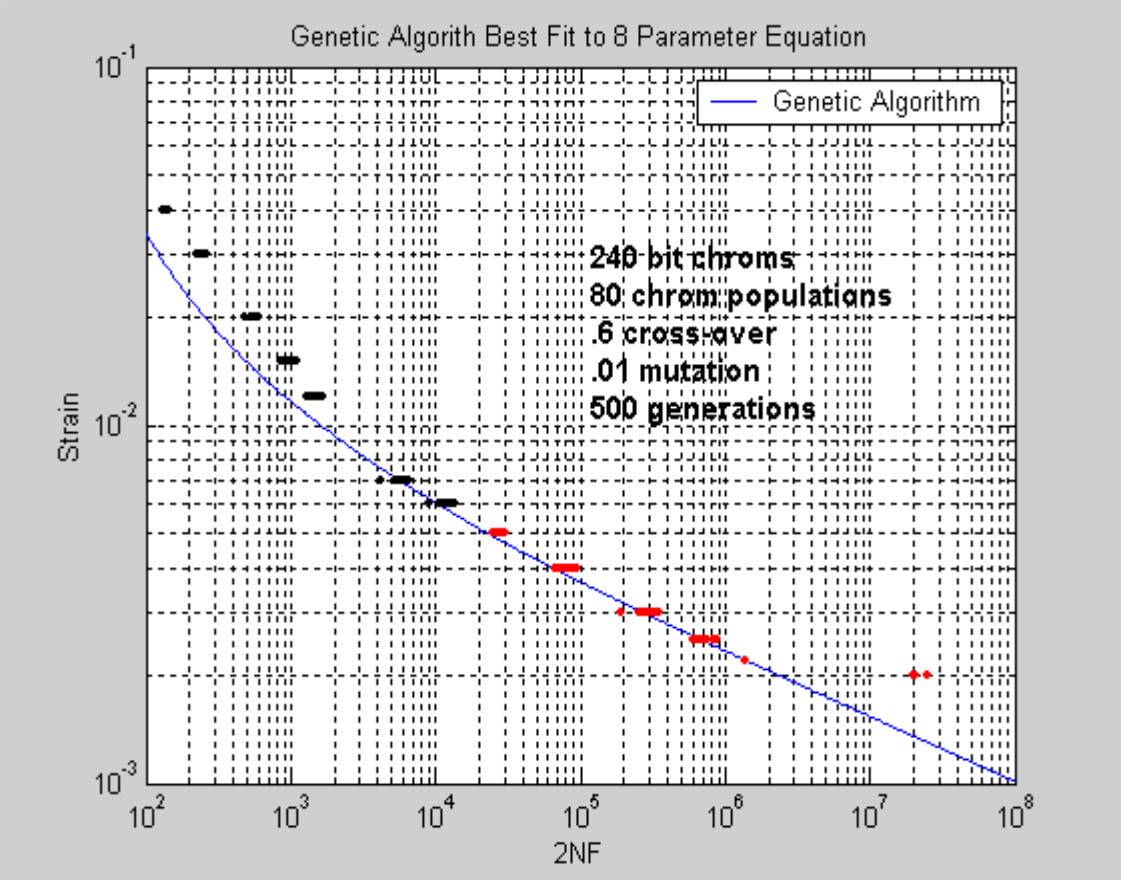

Figure. 30. Genetic Algorithm 8 Parameter Solution

Even with the complexity of this algorithm, the weakest link was the fact that there were difficulties making the solution fit all the data points. This was because a restrictive enough error function has yet to be determined. If a error function could be developed that would give equal importance to all data points in both location and slope, then the genetic algorithm could probably solve the 8 parameter equation. Another problem with the genetic algorithm was that the range of parameter constants was very important. If the range was too limited, then no solution was found. If the range was too wide, the solutions varied wildly. There was the additional problem of decimal values. As an example, if the selected range for chromosome mapping was 0 to 1, then even with a 30-bit string, the probability of finding a number between 0 and .1 is only 10%. If the number sought is less than .01 the probability drops to 1%. This problem exists even though the idea of genetic algorithms was to infuse an almost infinite number of

possibilities to the solution, in fact, there was a much higher likelihood of larger scaled numbers selection. The logic would say that if the investigator wants to find an answer between 0 and .01, then establish that as the range. Unfortunately, when that was done, the first problem presented itself and the range was too restrictive to allow a solution to be found. (One approach that may be used to address this problem is to use power-law scaling and mapping as opposed to a linear scaling used in the present implementation.)

An attempt was made to see if error minimization would continue through a large number of generations, even if there was a flaw with the fitness function.

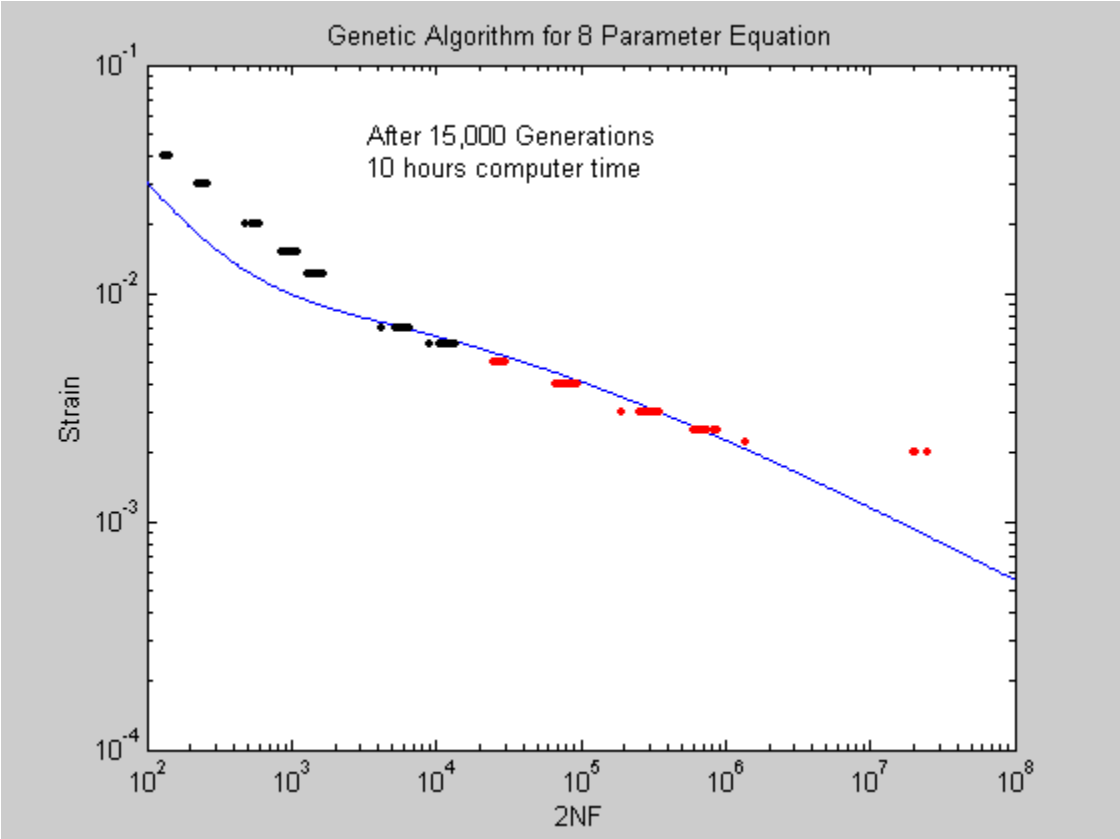

Figure. 31. Genetic Algorithm after 15,000 Generations

Unfortunately the solution at 500 generations was about the same as that done with 15000 generations. Clearly, the fitness function needs to be improved to drive the solution to match all mean test data points.

G. OTHER SOLUTION OPTIONS AND IDEAS

The previously mentioned methods represented only a few of the best possible solutions to this particular problem. Many other manipulations were tried with varying

degrees of success. A few will be mentioned here because the ideas could be of use to other problems.

Probably the most significant "other" method was also a simple one. Since neither the elastic data points nor the plastic data follow a straight line in log space throughout the range of life, a varying slope method would be an obvious option. Typically, the plastic data points follow a fairly linear trend until the intermediate range when the plastic strain drops off dramatically. This phenomenon significantly contributes to the knee of the total strain points. Plastic strain is not alone in its bad behavior. The elastic strain also exhibits a tendency to follow a line up until a point in which it also reduces significantly. If the problem is carefully broken down into smaller pieces and regressions are done across a smaller range of data, then the slopes of the equations could periodically be revised during the course of the probabilistic model. This method will provide the best approach,(other than the straight-line method!), to correcting for the knee problem.

Along the same idea as the previous method would be the use of several different equations for the material behavior. Each range (low, intermediate and high cycle) would have it's own equation. The only difference to the previous method is that this solution would have 3 independent solutions whereas the previous had the same solution with varying slopes and intercepts.

The genetic algorithm may have some use in finding the equation of a 4, 6 or 8 parameter equation in a specific region like the intermediate strain knee problem. Additionally, the use of a single Y intercept variable slope equation was explored. For this problem the rate of change of the slope was found to be log-linear.

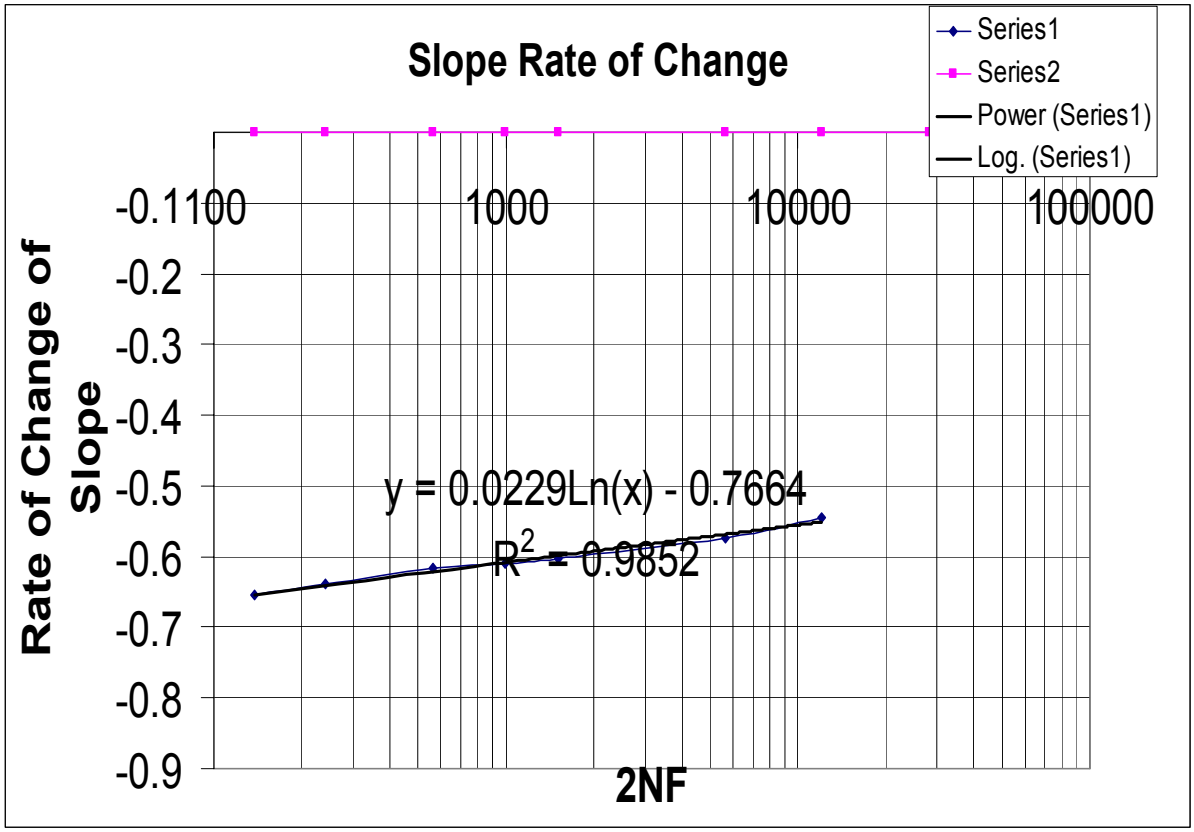

Figure. 32. Slope Rate of Change

This would imply that the following equation could be used to define the life.

$$\varepsilon_a = Y_{\text{int}} \times 2NF^{.0229Ln(2NF)-.7664}$$

The use of this equation may be a good fit for the low cycle region though the intermediate range. It could not be used though-out the entire life due to the nature of the natural logarithm function. The results are illustrated in the following figure.

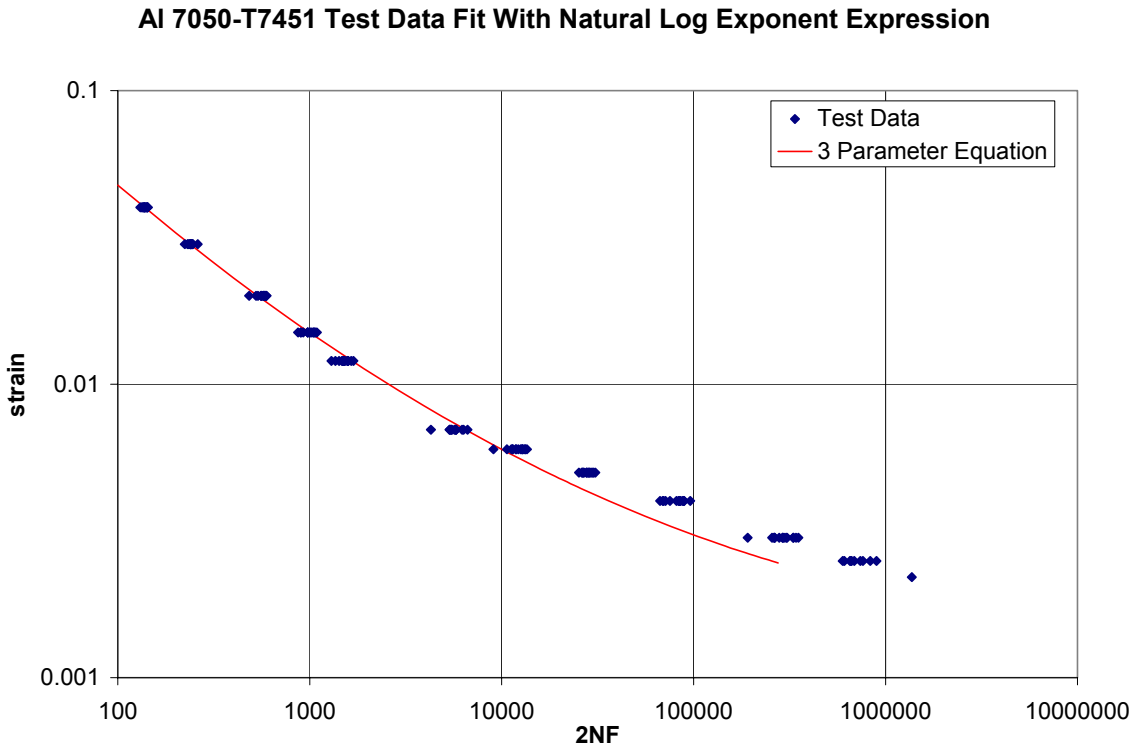

Figure. 33. 3 Parameter Natural Logarithmic Function

The use of this equation is limited. However, the rate of change until 2Nf is 10,000 is log-linear and the 3-parameter equation fits the data points very well until the 10,000 cycle range. An attempt was made with the use of genetic algorithms to find a better solution. However, without a perfected fitness function, the solutions obtained were not satisfactory.

VII. CONCLUSIONS

A. TESTING MUST PRODUCE CONCLUSIVE PROBABILITY DISTRIBUTION CHARACTERISTICS.

Without sufficient test data, it would be difficult to accurately develop a probabilistic strain-life model. Since the reason for such a probabilistic model is to stretch the range of life beyond traditional methods, the level of accuracy of the testing must be extremely high with a very high level of confidence. The test data for Al 7050-T7451 does not show any conclusive trends toward a probability distribution. Ultimately the testing may show that the data does exhibit a normal distribution. However, at this time, with the present test data, the investigator does not feel that it would be appropriate to assign a known probability distribution this data set if it is to be used in a subsequent life-extension analysis

B. VARIABLES CONTROLLING THE SCATTER MUST BE MEASURED AND QUANTIFIED AS THEY RELATE TO THE PROBABILITY DISTRIBUTION OF THE ACTUAL MATERIAL.

If the current testing is tightly controlled, the probability distribution may show a normal or log normal, or Weibull distribution, with a very small standard deviation. However, how does this idealized sample correspond to a real-world piece of metal in an actual aircraft component? To extend service life by inference to the probability distributions of idealized samples would lead to certain disaster. A comprehensive look must be made at all possibilities of the material variations that may be used to make aircraft components.

C. THE CENTRAL LIMIT THEOREM MAY NOT BE AN ACCURATE ASSUMPTION, DEPENDING ON HOW THE TESTING IS COMPLETED.

The normal distribution and the central limit theorem may not be an accurate assumption for current material fatigue test data sets. *With refined methods of testing methodology and material manufacturing, the degree of randomness required for a Gaussian distribution may no longer exist*. Imagine a random selection of people taken

near the exit of a NBA locker room. There would obviously be some pseudo-normal distribution with a very small standard deviation. The mean would be about 6'6''. Now imagine that a random sampling of people was taken of people coming out of the midget tent at the circus. Again, a normal distribution would characterize the population. This time the mean would be about 3'6''! Now imagine those two sample sets were grouped and used to describe the characteristics of a random population of all people in the world. Both sets were both drawn randomly. Would the total probability distribution demonstrate the central limit theorem? Obviously not! This is much the same as materials testing. If only NBA basketball stars are measured, but the world (of materials) includes midgets, then the probability model to extend life is flawed. Before this probability model can be completed, the investigators must understand how the midgets and the NBA stars will affect the total solution.

D. IT IS MUCH SIMPLER TO MODEL THE MONTE CARLO SIMULATION WITH USE OF THE 2NF SEEDING METHOD INSTEAD OF SOLVING FOR THE STRAIN AT EACH LEVEL.

The fact that the 4-parameter strain-life equation must be solved for with some sort of non-linear solver makes the 2NF seeding method much simpler to use. By placing many thousands of uniformly spaced 2NF data points into the simulation, determining the corresponding strain is simply a matter of evaluating the expression. This saves a significant amount of complexity and computational time. This method also does seem to be an extremely effective way of characterizing the nature of the total range of strain.

E. VARY ONLY THE 2 COEFFICIENTS OF THE STRAIN-LIFE EQUATION DURING MONTE CARLO SIMULATION.

Varying the exponents (b and c) of the strain-life equation invokes too much scatter to the probabilistic solution. The perturbation of the coefficients, $\sigma_f^{'}\ \varepsilon_f^{'}$, results in a profile that is sufficiently varied to characterize the scatter of the test data. For materials that exhibit strange variations, small variations of the exponents may help to model the behavior.

F. THE CORRELATION OF THE TEST SCATTER TO THE CONSTANT'S VARIATION, IS BEST DESCRIBED BY THE SIMPLEST METHOD.

There may be other formal methods that could describe the amount of perturbation required by the coefficients to model the probability distribution of fatigue data. However, the investigator found that simply varying the coefficients by the same percentage variation as the test scatter resulted in a satisfactory Monte Carlo simulation. Once a more complete set of testing has occurred, there may be a requirement to adjust this simple variation for more complex methods in order to predict more formal probability distributions. Currently, a uniform variation of constants is used. Accurate prediction of a normal distribution of test data may require the use of a normal variation of parameters.

G. IN ORDER TO CREATE A PROBABILISTIC MODEL OVER THE ENTIRE FATIGUE LIFE, A FUNCTION OR ALGORITHM MUST BE SPECIFIED THAT FITS ALL CHARACTERISTICS.

The "knee" and other irregularities cannot be fit with the use of the standard 4-parameter strain-life equation. If a probabilistic model is attempted, the data points near the "knee" or "the transient zone" will not lend themselves to be modeled accurately. The probabilistic model needs some sort of anchor point at each strain level about which a probability distribution may be defined.

H. AN EIGHT PARAMETER STRAIN-LIFE EQUATION MAY WORK TO SATISFACTORY FIT ALL CHARACTERISTICS OF FATIGUE LIFE TO INCLUDE THE "KNEE".

A strain-life equation that is the sum of 4 log-linear segments affords the flexibility to model the shape of the mean fatigue life with more accuracy than the 2-segment strain-life model. These 4 segments have no physical interpretation like elastic part and plastic part. However, the elastic and plastic equations can provide a baseline for estimation of all the 8 parameters.

I. FINDING THE RIGHT MIX OF THE 8 PARAMETERS IS EXTREMELY DIFFICULT.

The best method found was using a random variation of the exponents and then using a least squares-nonlinear solver to evaluate the corresponding coefficients until a satisfactory solution was found. As computationally advanced as this method was, it still was not better than the investigator's eye, patience and best guesses (namely, the heuristic approach).

J. GENETIC ALGORITHMS ARE AN IMPROVEMENT OVER RANDOM NUMBER SOLUTION FINDING TECHNIQUES, BUT THEIR ACCURACY IS, IN PART, FITNESS FUNCTION DRIVEN.

The manner in which a genetic algorithm seeks out the answer speeds up the solution process significantly over standard random number generation. However, better fitness function definitions must be developed before genetic algorithms can provide satisfactory solutions.

K. THE BEST MODELS MAY BE OBTAINED FROM THE SIMPLIEST METHODS.

The quickest and most accurate models were the simplest. Drawing a straight line between the mean data points from one level to the next, leaves no room for question. If enough strain levels are tested, these straight lines essentially define the curve.

Obviously, if only 2 strain levels (one low cycle and one high cycle) were tested then the straight line method would not predict the fatigue life at all, but how well would the data be represented anyway! Certainly, investigators could use the two levels to do a regression and then compute the elastic parts and plastic parts. Thus the stain-life equation could be used to create a curve of sorts that would theoretically describe the behavior. However, life prediction between the two points would not be realistic.

Does this mean that there should be tests taken at every strain level? (Certainly not.) The number of strain levels taken during these tests accurately describes the materials behavior in the region tested.

Another very simple model was the strain-level to strain-level, piece-wise solution of the 4-parameter strain-life equation. This algorithm added the ability for the final solution to have a degree of curvature. The slight curvature enhanced the accuracy of the final probabilistic model by rounding off the corners of the straight-line method.

Monte Carlo simulation of the 4, 6 or 8 parameters might make the testing evaluator feel more confident in his probabilistic model since it simulates the testing of tens of thousands or hundreds of thousands of samples. However, the best results will be obtained from connecting lines or curves, of confidence intervals from one strain level to the next as was done with the mean line fit.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. RECOMMENDATIONS

A. MORE TESTING IS THE STANDARD AND OBVIOUS ANSWER.

If aircraft must be extended beyond their current FLEs, then a very sound decision can only be made from sufficient evidence. If service lives are extended without enough testing, then the evaluators are betting against the very good intuition of many years of engineering experience and basing there assessment on un-conclusive data. More testing is definitely required.

Specifically, several strain levels should be selected in which to test a much larger number of samples. The investigator makes a guess that at least 25 samples will be required if a normal distribution does exist. The large number of samples is not necessary at all strain levels. 4%, 1% would be useful in characterizing the nature of low cycle fatigue. .9%,.7%,and .6% should be tested in large numbers to examine the probabilities associated with the "knee". One or two levels of lesser sample size should also be tested here to better characterize the shape of the knee. .04% and .02% could describe the nature of scatter in high cycle fatigue with a larger set of samples.

B. TESTING MUST ATTEMPT TO QUANTIFY THE VARIABILITY ASSOCIATED WITH CHANGES BETWEEN MANUFACTURES, GRAIN ORIENTATION, AND ANY OTHER MATERIAL CHARACTERISTICS THAT AFFECT LIFE.

Testing should include purposeful variations of the material in order to quantify the true nature of the actual aircraft construction materials. Testing methods should follow the ideas presented in Chapter II, section F.

C. A BETTER FITNESS FUNCTION MUST BE DEVELOPED THAT WILL ALLOW FOR A GENETIC ALGORITHM SOLUTION TO THE 8-PARAMETER EQUATION.

The 4-parameter strain-life equation may work for some materials if the nature of the "knee" is not too severe. Otherwise, an 8-parameter equation will sufficiently model and fit the data. Preliminary investigations revealed the difficult nature of the

uncertainties associated with fatigue data modeling. However, development of appropriate fitness functions reflecting the location and slope of the strain-life curve may result in satisfactory modeling of the fatigue life prediction using genetic algorithms.

D. TESTING MUST BE COMPLETED TO ESTABLISH CORRELATION BETWEEN % LOAD DROP AND .01 INCH MICRO-CRACK.

Since NAVAIR's Fatigue Life Expended (FLE) definition is based upon a 1 in 1000^{th} chance of a .01 inch micro-crack in existence, testing must establish at the % load drop to crack size relationship.

APPENDIX A(1) NAVAIR HCF DATA

NAVAIR HIGH CYCLE FATIGUE DATA			
Sample	Total Strain	2NF	Duplicate
HP1-45	0.005	25246	25246
HP1-25	0.005	26134	26134
HP1-56	0.005	26420	26420
HP1-33	0.005	26568	26568
HP1-6	0.005	26842	26842
HP1-57	0.005	27606	27606
HP1-29	0.005	27718	27718
HP1-14	0.005	28130	28130
HP1-21	0.005	28314	28314
HP1-3	0.005	28648	28648
HP1-17	0.005	28872	28872
HP1-41	0.005	29468	29468
HP1-52	0.005	29900	29900
HP1-12	0.005	30126	30126
HP1-40	0.005	30764	30764
HP1-43	0.004	66852	66852
HP1-15	0.004	68974	68974
HP1-7	0.004	69706	69706
HP1-53	0.004	71308	71308
HP1-26	0.004	75256	75256
HP1-31	0.004	81280	81280
HP1-2	0.004	83578	83578
HP1-50	0.004	84188	84188
HP1-24	0.004	84962	84962
HP1-38	0.004	85514	85514
HP1-36	0.004	87984	87984
HP1-46	0.004	88486	88486
HP1-9	0.004	88604	88604
HP1-19	0.004	89232	89232
HP1-58	0.004	95990	95990
HP1-8	0.003	191300	191300
HP1-39	0.003	256118	256118
HP1-32	0.003	261872	261872
HP1-18	0.003	264112	264112
HP1-11	0.003	264556	264556
HP1-28	0.003	279370	279370
HP1-1	0.003	291176	291176
HP1-60	0.003	291626	291626
HP1-22	0.003	295618	295618
HP1-54	0.003	303112	303112
HP1-35	0.003	307076	307076
HP1-47	0.003	329962	329962
HP1-13	0.003	331346	331346
HP1-49	0.003	341598	341598
HP1-44	0.003	351354	351354
HP1-27	0.0025	597166	597166
HP1-51	0.0025	609754	609754
HP1-42	0.0025	652596	652596
HP1-48	0.0025	660884	660884
HP1-23	0.0025	665700	665700
HP1-55	0.0025	686404	686404
HP1-30	0.0025	738994	738994
HP1-59	0.0025	762730	762730
HP1-37	0.0025	832536	832536
HP1-34	0.0025	895878	895878
HP1-61	0.0022	1370062	1370062
HP1-5	0.002	20153782	20153782
HP1-20	0.002	20167300	20167300
HP1-4	0.002	20467786	20467786
HP1-10	0.002	20494218	20494218
HP1-16	0.002	24657924	24657924

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A(2) METCUT LCF DATA

		METCUT LCF 2% Load Drop				
Test_Numbe	Max_Strain	LD	Segment	Elastic	Plastic	acheck
IPG2-2	0.006	0.02	11387	0.005694	0.000307	0.006001
IPG2-6	0.006	0.02	13211	0.005702	0.000276	0.005979
IPG2-8	0.006	0.02	11811	0.005781	0.000203	0.005984
IPG2-3	0.006	0.02	12203	0.005731	0.00022	0.005951
IPG2-10	0.006	0.02	12729	0.00571	0.000292	0.006002
IPG2-11	0.006	0.02	11921	0.005684	0.000314	0.005997
IPG2-14	0.006	0.02	13553	0.00571	0.000304	0.006014
IPG2-15	0.006	0.02	12547	0.005692	0.000315	0.006007
IPG2-17	0.006	0.02	9045	0.005727	0.000297	0.006024
IPG2-20	0.006	0.02	13257	0.005733	0.000249	0.005982
IPG2-22	0.006	0.02	11391	0.005636	0.000343	0.005979
IPG2-24	0.006	0.02	12875	0.005703	0.000302	0.006006
IPG2-25	0.006	0.02	12871	0.005677	0.00033	0.006007
IPG2-27	0.006	0.02	10649	0.005687	0.00031	0.005997
IPG2-29	0.006	0.02	11231	0.005652	0.000346	0.005998
IPG2-1	0.007	0.02	4277	0.006269	0.000711	0.00698
IPG2-5	0.007	0.02	5757	0.006212	0.000787	0.006999
IPG2-7	0.007	0.02	5793	0.006206	0.000758	0.006965
IPG2-9	0.007	0.02	6327	0.006279	0.000739	0.007018
IPG2-12	0.007	0.02	6217	0.00629	0.000727	0.007017
IPG2-13	0.007	0.02	5775	0.006258	0.000767	0.007025
IPG2-16	0.007	0.02	5357	0.006295	0.000731	0.007026
IPG2-18	0.007	0.02	5511	0.006242	0.000756	0.006998
IPG2-19	0.007	0.02	5465	0.006259	0.000707	0.006966
IPG2-21	0.007	0.02	5669	0.006252	0.000722	0.006974
IPG2-26	0.007	0.02	5413	0.006305	0.000687	0.006992
IPG2-28	0.007	0.02	6631	0.006233	0.000733	0.006966
IPG2-30	0.007	0.02	5345	0.006284	0.000723	0.007006
LPG2-3	0.012	0.02	1361	0.006701	0.005306	0.012007
LPG2-9	0.012	0.02	1489	0.006694	0.005291	0.011985
LPG2-11	0.012	0.02	1485	0.006655	0.005295	0.01195
LPG2-18	0.012	0.02	1569	0.006661	0.00529	0.011951
LPG2-21	0.012	0.02	1587	0.006652	0.005349	0.012001
LPG2-29	0.012	0.02	1471	0.006654	0.005338	0.011992
LPG2-33	0.012	0.02	1521	0.00667	0.005316	0.011985
LPG2-37	0.012	0.02	1643	0.006562	0.005418	0.01198
LPG2-42	0.012	0.02	1295	0.00674	0.005259	0.011999
LPG2-50	0.012	0.02	1581	0.006713	0.005297	0.01201
LPG2-52	0.012	0.02	1501	0.006668	0.005327	0.011995
LPG2-57	0.012	0.02	1517	0.006711	0.005295	0.012006
LPG2-65	0.012	0.02	1687	0.006651	0.005323	0.011974
LPG2-66	0.012	0.02	1573	0.006689	0.005293	0.011982
LPG2-75	0.012	0.02	1423	0.00664	0.005354	0.011994
LPG2-2	0.015	0.02	925	0.006859	0.00815	0.01501
LPG2-6	0.015	0.02	977	0.006806	0.008187	0.014992
LPG2-13	0.015	0.02	1059	0.006835	0.008154	0.014988
LPG2-19	0.015	0.02	1047	0.006781	0.00818	0.014962
LPG2-24	0.015	0.02	1005	0.00683	0.008168	0.014998

LPG2-26	0.015	0.02	869	0.006796	0.008191
LPG2-35	0.015	0.02	975	0.006839	0.008164
LPG2-40	0.015	0.02	997	0.00679	0.008195
LPG2-45	0.015	0.02	899	0.006831	0.008165
LPG2-49	0.015	0.02	977	0.006835	0.008164
LPG2-55	0.015	0.02	1067	0.006852	0.008153
LPG2-59	0.015	0.02	1089	0.006859	0.00813
LPG2-61	0.015	0.02	1023	0.006824	0.008168
LPG2-69	0.015	0.02	1043	0.006847	0.008138
LPG2-73	0.015	0.02	899	0.006755	0.008226
LPG2-5	0.02	0.02	483	0.00711	0.01293
LPG2-8	0.02	0.02	585	0.007036	0.012958
LPG2-15	0.02	0.02	585	0.007096	0.012898
LPG2-17	0.02	0.02	537	0.007139	0.012897
LPG2-25	0.02	0.02	561	0.007018	0.012977
LPG2-30	0.02	0.02	559	0.007068	0.012931
LPG2-32	0.02	0.02	593	0.007046	0.012925
LPG2-39	0.02	0.02	571	0.007097	0.012862
LPG2-41	0.02	0.02	577	0.007091	0.012892
LPG2-48	0.02	0.02	525	0.007056	0.012934
LPG2-51	0.02	0.02	575	0.00715	0.012858
LPG2-58	0.02	0.02	589	0.007043	0.012945
LPG2-63	0.02	0.02	595	0.007093	0.012878
LPG2-70	0.02	0.02	579	0.007085	0.012894
LPG2-71	0.02	0.02	557	0.007066	0.012906
LPG2-4	0.03	0.02	223	0.007644	0.022352
LPG2-10	0.03	0.02	239	0.007562	0.022434
LPG2-12	0.03	0.02	246	0.007548	0.022388
LPG2-20	0.03	0.02	239	0.007599	0.022396
LPG2-23	0.03	0.02	233	0.007541	0.022453
LPG2-27	0.03	0.02	261	0.007549	0.022439
LPG2-31	0.03	0.02	237	0.007569	0.022421
LPG2-36	0.03	0.02	241	0.007511	0.022453
LPG2-44	0.03	0.02	239	0.007571	0.022427
LPG2-46	0.03	0.02	241	0.00755	0.022444
LPG2-60	0.03	0.02	231	0.007539	0.022449
LPG2-64	0.03	0.02	239	0.007483	0.022473
LPG2-68	0.03	0.02	243	0.007611	0.022396
LPG2-74	0.03	0.02	245	0.007572	0.022415
LPG2-1	0.04	0.02	133	0.0079	0.032095
LPG2-7	0.04	0.02	137	0.007853	0.032117
LPG2-14	0.04	0.02	143	0.007872	0.031976
LPG2-16	0.04	0.02	139	0.007865	0.032074
LPG2-22	0.04	0.02	131	0.007798	0.032174
LPG2-28	0.04	0.02	137	0.007892	0.032108
LPG2-34	0.04	0.02	141	0.007855	0.032142
LPG2-38	0.04	0.02	137	0.00784	0.032138
LPG2-43	0.04	0.02	143	0.007859	0.032109
LPG2-47	0.04	0.02	137	0.007925	0.032072
LPG2-53	0.04	0.02	143	0.007918	0.032075
LPG2-56	0.04	0.02	135	0.007831	0.032159
LPG2-62	0.04	0.02	139	0.007863	0.032127
LPG2-67	0.04	0.02	135	0.007916	0.032083
LPG2-72	0.04	0.02	141	0.007896	0.032097

APPENDIX B. STRAINLIFE9.M


```
%       Script File 'Strainlife9.m'
%       Thomas Heffern
%       17 July 2002   Modified 23 August 2002
%   This program generates many random strain life data points
%   based on experimental data from excel spreadsheet "newtestdatainputs"
% ____________________________________________________________________________
_____________-_____________________________
clear
format short
global FSC FDC B1 C1 E1;
%----------------------------------------------------------------------------
-----------------------------------------------
%USER INPUTS
N = input ('Enter the number of calculations at each NF level, 1-100, N = '
);
lowest = input ('Enter the power of lowest strain level desired, ie. .0001
would = -2.5 (recommended!), lowest =    ');
highest = input ('Enter the power of the highest strain level desired, ie.
 .1 would = -1,  highest =    ');
%----------------------------------------------------------------------------
-----------------------------------------------
% import of test data groupings and data analysis from Test Data Inputs Exc
el worksheet
grp1 = xlsread('newtestdatainputs','.006');
grp2 = xlsread('newtestdatainputs','.007');
grp3 = xlsread('newtestdatainputs','.012');
grp4 = xlsread('newtestdatainputs','.015');
grp5 = xlsread('newtestdatainputs','.02');
grp6 = xlsread('newtestdatainputs','.03');
grp7 = xlsread('newtestdatainputs','.04');
%----------------------------------------------------------------------------
-----------------------------------------------
% TEST DATA INPUTS
%   Input the basic constants, Data is read from excel spread sheet ' new t
est data inputs'
%  The constants have previously been computed in the excel file using a lo
g-linear regression fit of the data
%
A = xlsread('newtestdatainputs','MATLABINPUTVECTOR');        %NOTE THIS IS
A COLUMN VECTOR
fsc = A(1,1);           %fatigue strength coeff
fdc = A(2,1);           %fatigue ductility coeff
b = A(5,1);             %fatigue strength exponent
c = A(8,1);             %fatigue ductility exponent
E = A(9,1);             % E modulus of elasticity
%
testNFmean = [mean(grp1(:,3)),mean(grp2(:,3)),mean(grp3(:,3)),mean(grp4(:,3
)),mean(grp5(:,3)),mean(grp6(:,3)),mean(grp7(:,3))];
testNFstd = [std(grp1(:,3)),std(grp2(:,3)),std(grp3(:,3)),std(grp4(:,3)),st
```

```
d(grp5(:,3)),std(grp6(:,3)),std(grp7(:,3))];
% using the uniform distribution model
testrange = [range(grp1(:,3)),range(grp2(:,3)),range(grp3(:,3)),range(grp4(:,3)),range(grp5(:,3)),range(grp6(:,3)),range(grp7(:,3))];
datavar = (testrange./2)./testNFmean;
meandatavar = mean(datavar);
%-------------------------------------------------------------------------------------------
% Seeding the space with 2NF calculations
nf = logspace(1,2);
NF = [nf';nf'*10;nf'*100;nf'*1000;nf'*10000;nf'*100000];
NFsize = length(NF);
for i = 1:N;
    NFmatrix(:,i) = NF;
end

% IF ALL 4 CONSTANTS ARE VARIED SAME AS DATA's VARIATION
strain = ((unifrnd((fsc-(meandatavar/1)*fsc),(fsc+(meandatavar/1)*fsc),NFsize,N)./E).*(NFmatrix).^(unifrnd((b-(meandatavar/1)*b),(b+(meandatavar/1)*b),NFsize,N)) + ...
    (unifrnd((fdc-(meandatavar/1)*fdc),(fdc+(meandatavar/1)*fdc),NFsize,N)).*(NFmatrix).^(unifrnd((c-(meandatavar/1)*c),(c+(meandatavar/1)*c),NFsize,N)) );

loglog(NFmatrix,strain,'y.' )   % A rough, unsorted view of all data points
%hold on
pause(1)
%-------------------------------------------------------------------------------------------
%Now all those random data points need sorting
% first make a vector of the data intervals in logspace
y = logspace(lowest,highest,51);
count = 0;
empty = 0;
for lvl = 1:50;
    [I,J] = find(strain>y(lvl) & strain<=y(lvl+1));
    tempstrn= strain(I,J);
    sortedstrain = tempstrn(:);                     %sorted strain level in column
    branch = isempty(sortedstrain);
    if branch == 1
        empty = empty+1;
    elseif branch == 0;
        count = count + 1;
        tempNF = NFmatrix(I,J);
        sortedNF = tempNF(:);                       %corresponding NFs in column
        sortmeanNF(count,1) = mean(sortedNF);
        sortmaxNF(count,1) = max(sortedNF);
```

```
        sortminNF(count,1) = min(sortedNF);
        sortmeanstrain(count,1) = mean(sortedstrain);

    end

end
%-------------------------------------------------------------------------↙
----------------------------------------
% PLOTTING THE SIMULATED STRAIN LIFE DATA
figure
loglog(sortmeanNF,sortmeanstrain,'b',sortmaxNF,sortmeanstrain,'b',sortminNF↙
,sortmeanstrain,'b');
hold on
grid
title('Variational Strainlife with Actual Test Data')
xlabel('2NF')
ylabel('Strain')

%-------------------------------------------------------------------------↙
----------------------------------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME FROM C↙
OMPANION EXCEL WORK SHEET 'NEW TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY
    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);

    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')
    legend('simulated mean', 'simulated min', 'simulated max', 'METCUT LCF ↙
data', 'NAVAIR HCF data')
%-------------------------------------------------------------------------↙
----------------------------------------
% PLOTTING BASIC STRAIN LIFE LINE (COFFIN-MANSON) FOR COMPARISON

    NF2 = [10:100:max(NF)];
    stnlife = ((fsc/E).*(NF2).^b)+fdc.*(NF2).^c;
    loglog(NF2,stnlife,'k')
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. STRAINLIFE8A.M


```
%       Script File 'Strainlife8a.m'
%       Thomas Heffern
%       18 May 2002, revised 10 July, 12 July,
%       REVISED FOR PERFORMANCE AND NEW DATA on 23 August 2002
%   This program generates many random strain life data points
%   based on inputed experimental data from excel spreadsheet
%   " newtest data inputs"
%   2NF is solved by using the fzero command
%   at various levels of strain...CAUTION...it runs very slowly

%_____________________________________________________________________
clear
format short
global FSC FDC B1 C1 E1;

%---------------------------------------------------------------------
%USER INPUTS
%   Define the random number permutations, 100 calculations in a given rang↙
e should be plenty!
%   calculations greater than 500 may cause significant time (greater than ↙
24 hours!)
N = input ('Enter the number of calculations at each strain level, 1-100, N↙
 = ');
lowest = input ('Enter the power of lowest strain level desired, ie. .0001 ↙
would = -2.5 (recommended!), lowest =    ');
highest = input ('Enter the power of the highest strain level desired, ie. ↙
 .1 would = -1,  highest =    ');
strain = logspace(highest,lowest);
SLnum = length(strain);
%---------------------------------------------------------------------
% import of test data groupings and data analysis from Test Data Inputs Exc↙
el worksheet
grp1 = xlsread('newtestdatainputs','.006');
grp2 = xlsread('newtestdatainputs','.007');
grp3 = xlsread('newtestdatainputs','.012');
grp4 = xlsread('newtestdatainputs','.015');
grp5 = xlsread('newtestdatainputs','.02');
grp6 = xlsread('newtestdatainputs','.03');
grp7 = xlsread('newtestdatainputs','.04');
%---------------------------------------------------------------------
% TEST DATA INPUTS
%   Input the basic constants, Data is read from excel spread sheet 'test d↙
ata inputs'
%
A = xlsread('newtestdatainputs','MATLABINPUTVECTOR');        %NOTE THIS IS ↙
A COLUMN VECTOR
fsc = A(1,1);           %fatigue strength coeff

fdc = A(2,1);           %fatigue ductility coeff
```

```
b = A(5,1);             %fatigue strength exponent

c = A(8,1);             %fatigue ductility exponent

E = A(9,1);             % E modulus of elasticity


testNFmean = [mean(grp1(:,3)),mean(grp2(:,3)),mean(grp3(:,3)),mean(grp4(:,3↙
)),...
        mean(grp5(:,3)),mean(grp6(:,3)),mean(grp7(:,3))];
testNFstd = [std(grp1(:,3)),std(grp2(:,3)),std(grp3(:,3)),std(grp4(:,3)),..↙
.
        std(grp5(:,3)),std(grp6(:,3)),std(grp7(:,3))];
% using the uniform distribution model
testrange = [range(grp1(:,3)),range(grp2(:,3)),range(grp3(:,3)),range(grp4(↙
:,3)),...
        range(grp5(:,3)),range(grp6(:,3)),range(grp7(:,3))];
datavar = (testrange./2)./testNFmean;
meandatavar = mean(datavar);

for level = 1:SLnum;
    strnlevel =  strain(:,level)        % this helps user to monitor progre↙
ss.
    % This loop computes new random constants with a normal distribution an↙
d then solves for the
    % value of 2NF in the strainlife equation using the fzero function
    % unfortunately the solver can only handle scalars, so this loop eats u↙
p a lot of time

    for iter = 1:N;

         FSC = unifrnd((fsc-(meandatavar/1)*fsc),(fsc+(meandatavar/1)*fsc))↙
;
         B1 = unifrnd((b-(meandatavar/1)*b),(b+(meandatavar/1)*b));
         FDC = unifrnd((fdc-(meandatavar/1)*fdc),(fdc+(meandatavar/1)*fdc))↙
;
         C1 = unifrnd((c-(meandatavar/1)*c),(c+(meandatavar/1)*c));

        % I was having trouble getting these variables into the function..
        % the 'save', then 'load' (in the function) commands works well.
        save strainlifevars
        %
        NF(iter,level) = fzero(@strnliffcn,[1, 10000000000000000000]);
        % strain2 is created to ease in plotting vector creation
        strain2(iter,level) = strnlevel;

    end     % of the iteration loop
```

```
end          % of the strain levels loop

pause(2)
format short e

% CREATION OF A USEFUL MATRIX OF STRAIN LEVELS AND CORRESPONDING 2NF
LIFE = [strain;NF];
%-------------------------------------------------------------------------↙
----------
% PLOTTING THE SIMULATED STRAIN LIFE DATA

% THE EASY WAY of plotting all the simulated data points
    loglog(NF,strain2)
    grid
    title('Variational Strainlife with Actual Test Data')
    xlabel('2NF')
    ylabel('Strain')
    hold on
% Computation of the means and standard deviations at each simulated strain↙
 level

for i = 1:SLnum;
    meanNF(:,i) = mean(NF(:,i));
    maxNF(:,i) = max(NF(:,i));
    minNF(:,i) = min(NF(:,i));

end
    loglog(meanNF,strain,'b')
    loglog(minNF,strain,'r',maxNF,strain,'r')

%-------------------------------------------------------------------------↙
----------
% PLOTTING THE TEST DATA AND A PLOT OF THE BASIC STRAIN LIFE LINE AND STAND↙
ARD DEVIATIONS

    NF2 = [10:100:max(NF(:,level))];
    stnlife = ((fsc/E).*(NF2).^b)+fdc.*(NF2).^c;
    stnlifestdleft = (NF2-(stdnum*datavar).*NF2);
    stnlifestdrite = (NF2+(stdnum*datavar).*NF2);
    loglog(NF2,stnlife,'k')
    loglog(stnlifestdrite,stnlife,'c',stnlifestdleft,stnlife,'c')
    %
    pause(10)
    %-------------------------------------------------------------------------↙
-------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME FROM C↙
OMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY
```

```
act2NF = xlsread('newtestdatainputs','2NFvector');
actstrain = xlsread('newtestdatainputs','Strainvector');
HCF = xlsread('newtestdatainputs','NAVHCF');
HCF2NF = HCF(:,3);
HCFstrn = HCF(:,1);

loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')
```

APPENDIX D. STRAINLIFELINZ.M


```
% Strainlifelinz   just straight lines connecting the points
strain = [.04,.03,.02,.015,.012,.007,.006,.005,.004,.003,.0025,.002];
NF = [138,239.8,565,990,1514,5657,12045,28050,81461,290680,710264,21188202]↙
;
powers = log10(NF);
std = [.026,.035,.054,.068,.068,.102,.099,.057,.109,.14,.13.6];
min = [131,223,483,869,1295,4277,9045,25246,66852,191300,597166,20153782];
powers1 = log10(min);
max = [143,261,595,1089,1687,6631,13553,30764,95990,351354,895878,24657924]↙
;
powers2 = log10(max);

points = (length(NF)-1);
for i = 1:points;

b = (log10(strain(i))-log10(strain(i+1)))/(log10(NF(i))-log10(NF(i+1)));
int = strain(i)/(NF(i)^b);
NFvect = logspace(powers(i),powers(i+1));
stnvect = int.*NFvect.^b;
%
b1 = (log10(strain(i))-log10(strain(i+1)))/(log10(min(i))-log10(min(i+1)));
int1 = strain(i)/(min(i)^b1);
minvect = logspace(powers1(i),powers1(i+1));
stn1vect = int1.*minvect.^b1;
%
b2 = (log10(strain(i))-log10(strain(i+1)))/(log10(max(i))-log10(max(i+1)));
int2 = strain(i)/(min(i)^b2);
maxvect = logspace(powers2(i),powers2(i+1));
stn2vect = int2.*minvect.^b2;

loglog(NFvect,stnvect,'k',minvect,stn1vect,'r',maxvect,stn2vect,'r')
hold on
pause(.2)
end
grid
%------------------------------------------------------------------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME FROM
% COMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY

    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);

    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')
    legend('mean','min','max','LCF','HCF')
    title('Linear Solution of Probability Profile')
```



```
    xlabel('2NF')
    ylabel('strain')
 break
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. STRAINLIFEGA.M


```
% StrainlifeGA  Genetics of point two defined by point one, both normal and
 uniform disb sims
clear
strain = [.04,.03,.02,.015,.012,.007,.006,.005,.004,.003,.0025,.002];
NF = [138,239.8,565,990,1514,5657,12045,28050,81461,290680,710264,21188202]
;
minNF = [131,223,483,869,1295,4277,9045,25246,66852,191300,597166,20153782]
;
maxNF = [143,261,595,1089,1687,6631,13553,30764,95990,351354,895878,2465792
4];
minperct = minNF./NF;
maxperct = maxNF./NF;
correction = 0;
for i = 1:11
    % note the %ed out instructions correspond to a normal variation of par
ameters
    fsc = .015 +(.5*.015) * randn(1000000,1);
    %fsc = .011 +(.0229-.011) * rand(1000000,1);
    b = -.05 + (.5*.05) * randn(1000000,1);
    %b = -.068 + (-.158+.068) * rand(1000000,1);
    fdc = 1.2418 + (.5*1.2418) *randn(1000000,1);
    %fdc = 1.248 + (10 -1.248) *rand(1000000,1);
    c = -.7337 + (.5*.7337) * randn(1000000,1);
    %c = -.7337 + (-1.1+.7337) * rand(1000000,1);
    straintest = fsc.*NF(i).^b +fdc.*NF(i).^c;   % note fsc really is the i
ntercept, fsc/E
    diff = abs(strain(i)-straintest);
    [I] = find(diff < .00001  );
    fscgenes = fsc(I);
    bgenes = b(I);
    fdcgenes = fdc(I);
    cgenes = c(I);
    straintest(I);
    NFnext = NF(i+1);
    nextstrain = fscgenes.*NFnext.^bgenes + fdcgenes.*NFnext.^cgenes;
    diff2 = abs(strain(i+1)-nextstrain);
    x = min(diff2);

    [I2] = find(diff2 == x);
    strain(i+1)
    nextstrain(I2)
    NFvec = [NF(i):10:NF(i+1)];
    strainvect = fscgenes(I2).*NFvec.^bgenes(I2) + fdcgenes(I2).*NFvec.^cge
nes(I2);
    %saving the coeffs for later use
    COFFs(i,:) = [fscgenes(I2),bgenes(I2),fdcgenes(I2),cgenes(I2)];
    %
    NFvecmin = NFvec.*(mean(minperct)-correction);
    NFvecmax = NFvec.*(mean(maxperct)+correction);
```

```
    correction = correction +.02;
    % PLOT OF SCATTER FACTOR = 2
    scatfact2 = NFvec./2;
    loglog(NFvec,strainvect, NFvecmin,strainvect,'r',NFvecmax,strainvect,'r↙
', scatfact2,strainvect,'k')
    grid
    hold on
    pause(1)
end
%-------------------------------------------------------------------------↙
----------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME FROM
% COMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY

    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);

    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')
    title('4 Parameter Evolutionary Algorithm Strain-Life')
    xlabel('2NF')
    ylabel('Strain')
    legend('Mean Strain','Min Strain','Max Strain','METCUT LCF','NAVAIR HCF↙
','Scatter Factor = 2')

 %=========================================================================↙
============
 %  MONTE CARLO SIMULATION
 contin = input('Do you want to run Monte Carlo Simulation? 1 == yes, 5== n↙
o     ');
 if contin ==1


 %-------------------------------------------------------------------------↙
-----------
%USER INPUTS
N = input ('Enter the number of calculations at each NF level, 1-100, N = '↙
);
%P = input ('Enter the power of 10 for the max 2NF, P =    ');
lowest = input ('Enter the power of lowest strain level desired, ie. .0001 ↙
would = -2.5 (recommended!), lowest =    ');
highest = input ('Enter the power of the highest strain level desired, ie. ↙
 .1 would = -1,  highest =    ');
%stdnum = input('Enter the number of standard deviations to plot, 1,2 or 3.↙
.stdnum =   ');
```

```
%-------------------------------------------------------------------------↙
-----------
%import of test data groupings and data analysis from Test Data Inputs Exce↙
l worksheet
grp1 = xlsread('newtestdatainputs','.006');
grp2 = xlsread('newtestdatainputs','.007');
grp3 = xlsread('newtestdatainputs','.012');
grp4 = xlsread('newtestdatainputs','.015');
grp5 = xlsread('newtestdatainputs','.02');
grp6 = xlsread('newtestdatainputs','.03');
grp7 = xlsread('newtestdatainputs','.04');
%-------------------------------------------------------------------------↙
----------------
% TEST DATA INPUTS
%   Input the basic constants, Data is read from excel spread sheet 'new te↙
st data inputs'

testNFmean = [mean(grp1(:,3)),mean(grp2(:,3)),mean(grp3(:,3)),mean(grp4(:,3↙
)),mean(grp5(:,3)),mean(grp6(:,3)),mean(grp7(:,3))];
testNFstd = [std(grp1(:,3)),std(grp2(:,3)),std(grp3(:,3)),std(grp4(:,3)),st↙
d(grp5(:,3)),std(grp6(:,3)),std(grp7(:,3))];
%datavar = mean(testNFstd./testNFmean);    % for use with normal distributi↙
on model
% using the uniform distribution model
testrange = [range(grp1(:,3)),range(grp2(:,3)),range(grp3(:,3)),range(grp4(↙
:,3)),range(grp5(:,3)),range(grp6(:,3)),range(grp7(:,3))];
datavar = (testrange./2)./testNFmean;
meandatavar = mean(datavar);
%-------------------------------------------------------------------------↙
----------------
% Alternate method of seeding the space with 2NF calculations
figure
powers = log10(NF);
j = 1;
for j = 1:11

    NF = logspace(powers(j),powers(j+1))';
    NFsize = length(NF);
    i = 1;
    for i = 1:N;
        NFmatrix(:,i) = NF;
    end
    mdv = meandatavar;
    A = COFFs(j,1);
    B = COFFs(j,2);
    C = COFFs(j,3);
    D = COFFs(j,4);
    %for use with average variation model
    strainrnd = (unifrnd((A-(mdv/2)*A),(A+(mdv/2)*A),NFsize,N)).*(NFmatrix)↙
```

```
.^unifrnd((B-(mdv/2)*B),(B+(mdv/2)*B),NFsize,N)...
    + (unifrnd((C-(mdv/2)*C),(C+(mdv/2)*C),NFsize,N)).*(NFmatrix).^unifrnd(↙
(D-(mdv/2)*D),(D+(mdv/2)*D),NFsize,N);

    start = [1,51,101,151,201,251,301,351,401,451,501];
    START = start(j);
    stop = [50,100,150,200,250,300,350,400,450,500,550];
    STOP = stop(j)
    strainrand(START:STOP,:) = strainrnd;
    NFmatrixtot(START:STOP,:) = NFmatrix;
end

%-------------------------------------------------------------------------↙
------------
%Now all those random data points need sorting
% first make a vector of the data intervals in logspace
y = logspace(lowest,highest,51);
count = 0;
empty = 0;
for lvl = 1:50;
    [I,J] = find(strainrand>y(lvl) & strainrand<=y(lvl+1));
    tempstrn= strainrand(I,J);
    sortedstrain = tempstrn(:);                     %sorted strain level in↙
 column
    branch = isempty(sortedstrain);
    if branch == 1
        empty = empty+1;

    elseif branch == 0;
        count = count + 1;
    %if sortedstrain ~= []; %not equal to
        tempNF = NFmatrixtot(I,J);
        sortedNF = tempNF(:);                       %corresponding NFs in c↙
olumn
        sortmeanNF(count,1) = mean(sortedNF);
        sortmaxNF(count,1) = max(sortedNF);
        sortminNF(count,1) = min(sortedNF);
        sortmeanstrain(count,1) = mean(sortedstrain);

    end

end
%-------------------------------------------------------------------------↙
---------------
% PLOTTING THE SIMULATED STRAIN LIFE DATA
loglog(sortmeanNF,sortmeanstrain,'b',sortmaxNF,sortmeanstrain,'b',sortminNF↙
,sortmeanstrain,'b');
hold on
    grid
```

```
    title('Variational Strainlife with Actual Test Data')
    xlabel('2NF')
    ylabel('Strain')



%-------------------------------------------------------------------------
-------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME FROM
% COMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY

    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);
    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. STRAINFIT.M


```
% strainfit.m
% plot the mid points
clear
strain = [.04,.03,.02,.015,.012,.007,.006,.005,.004,.003,.0025,.002];
NF = [138,239.8,565,990,1514,5657,12045,28050,81461,290680,710264,21188202]↙
;
loglog(NF,strain,'r')
grid
hold on
%-----------------------------------------------------------------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME
% FROM COMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY

    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);

    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')

 %_____________________________________________________________________
 % Least squares curve fit of the data
 normerror = 5;
 count = 1;
 i = 0;
 while normerror >= .001
     i = i+1;
    x0 = [.1 .1 .1  .1];
    ub = [100 100 100 100];
    lb = [.00001 .00001 .00001 .00001];
     ex1(i,1) = unifrnd((-2.5),(.5));
     ex2(i,1) = unifrnd((-2.5),(.5));
     ex3(i,1) = unifrnd((-2.5),(.5));
     ex4(i,1) = unifrnd((-2.5),(.5));
    options = optimset('LineSearchtype','quadcubic');
     F = lsqcurvefit(@lifefcn2,x0,act2NF,actstrain,lb,ub,[options],ex1(i,1)↙
,ex2(i,1),ex3(i,1),ex4(i,1));
     Coef(i,:) = F;
     strnfit = F(1).*NF.^ex1(i,1) + F(2).*NF.^ex2(i,1) +F(3).*NF.^ex3(i,1) ↙
+ F(4).*NF.^ex4(i,1);
     error = abs(strnfit-strain)./strain;
     error(6) = 1000*error(6);         %the error is weighted with trial in↙
 error to get best solution
     error(9) = 900*error(9);
     error(10) = 300*error(10);
     error(11) = 400*error(11);
     error(12) = 1000*error(12);
```

```
     normerror = norm(error);
     normerrors(i,:) = normerror;
     count = count+1
        if count == 1000
            normerror = .000000000000001;
        end
end


 [E,I] = sort(normerrors);
 strnfitbest = Coef(I(1),1).*NF.^ex1(I(1),1) + Coef(I(1),2).*NF.^ex2(I(1),1↙
) +Coef(I(1),3).*NF.^ex3(I(1),1) + Coef(I(1),4).*NF.^ex4(I(1),1);

 loglog(NF,strnfitbest,'k')
 % now the Monte Carlo simulation
 A = Coef(I(1),1);
 B = ex1(I(1),1);

 C = Coef(I(1),2);
 D = ex2(I(1),1);

 E = Coef(I(1),3);
 F = ex3(I(1),1);

 G = Coef(I(1),4);
 H = ex4(I(1),1);

 %contin = input('Do you want Monte Carlo simulation? 1 == yes       5 == n↙
o')
 contin =1;
 if contin ==1
 %-------------------------------------------------------------------------
%USER INPUTS
N = input ('Enter the number of calculations at each NF level, 1-100, N = '↙
);
%P = input ('Enter the power of 10 for the max 2NF, P =    ');
lowest = input ('Enter the power of lowest strain level desired, ie. .0001 ↙
would = -2.5 (recommended!), lowest =    ');
highest = input ('Enter the power of the highest strain level desired, ie. ↙
 .1 would = -1,  highest =    ');
%stdnum = input('Enter the number of standard deviations to plot, 1,2 or 3.↙
.stdnum =   ');

%-------------------------------------------------------------------------↙
-
% import of test data groupings and data analysis from Test Data Inputs Exc↙
el worksheet
grp1 = xlsread('newtestdatainputs','.006');
grp2 = xlsread('newtestdatainputs','.007');
```

```
grp3 = xlsread('newtestdatainputs','.012');
grp4 = xlsread('newtestdatainputs','.015');
grp5 = xlsread('newtestdatainputs','.02');
grp6 = xlsread('newtestdatainputs','.03');
grp7 = xlsread('newtestdatainputs','.04');
%-------------------------------------------------------------------------
% TEST DATA INPUTS
%
testNFmean = [mean(grp1(:,3)),mean(grp2(:,3)),mean(grp3(:,3)),mean(grp4(:,3↙
)),mean(grp5(:,3)),mean(grp6(:,3)),mean(grp7(:,3))];
testNFstd = [std(grp1(:,3)),std(grp2(:,3)),std(grp3(:,3)),std(grp4(:,3)),st↙
d(grp5(:,3)),std(grp6(:,3)),std(grp7(:,3))];
%datavar = mean(testNFstd./testNFmean);    % for use with normal distributi↙
on model
% using the uniform distribution model
testrange = [range(grp1(:,3)),range(grp2(:,3)),range(grp3(:,3)),range(grp4(↙
:,3)),range(grp5(:,3)),range(grp6(:,3)),range(grp7(:,3))];
datavar = (testrange./2)./testNFmean;
meandatavar = mean(datavar);
%-------------------------------------------------------------------------↙
-
% Alternate method of seeding the space with 2NF calculations
nf = logspace(1,2);
NF = [nf';nf'*10;nf'*100;nf'*1000;nf'*10000;nf'*100000];
NFsize = length(NF);
for i = 1:N;
    NFmatrix(:,i) = NF;
end
mdv = meandatavar;
%for use with average variation model
strainrand = (unifrnd((A-(mdv/2)*A),(A+(mdv/2)*A),NFsize,N)).*(NFmatrix).^B↙
 + (unifrnd((C-(mdv/2)*C),(C+(mdv/2)*C),NFsize,N)).*(NFmatrix).^D + ...
         (unifrnd((E-(mdv/2)*E),(E+(mdv/2)*E),NFsize,N)).*(NFmatrix).^F + (↙
unifrnd((G-(mdv/2)*G),(G+(mdv/2)*G),NFsize,N)).*(NFmatrix).^H;

figure
loglog(NFmatrix,strainrand,'y.' )
%hold on
pause(.1)

%-------------------------------------------------------------------------↙
-----------------------------------------------
%Now all those random data points need sorting
% first make a vector of the data intervals in logspace
y = logspace(lowest,highest,51);
count = 0;
empty = 0;
for lvl = 1:50;
    [I,J] = find(strainrand>y(lvl) & strainrand<=y(lvl+1));
```

```
    tempstrn= strainrand(I,J);
    sortedstrain = tempstrn(:);                        %sorted strain level in↙
 column
    branch = isempty(sortedstrain);
    if branch == 1
        empty = empty+1;

    elseif branch == 0;
        count = count + 1;
    %if sortedstrain ~= []; %not equal to
        tempNF = NFmatrix(I,J);
        sortedNF = tempNF(:);                          %corresponding NFs in c↙
olumn
        sortmeanNF(count,1) = mean(sortedNF);
        sortmaxNF(count,1) = max(sortedNF);
        sortminNF(count,1) = min(sortedNF);
        sortmeanstrain(count,1) = mean(sortedstrain);

    end

end
%-------------------------------------------------------------------------↙
-----------------
% PLOTTING THE SIMULATED STRAIN LIFE DATA
    figure
loglog(sortmeanNF,sortmeanstrain,'b',sortmaxNF,sortmeanstrain,'b',sortminNF↙
,sortmeanstrain,'b');
hold on

    grid
    title('Variational Strainlife with Actual Test Data')
    xlabel('2NF')
    ylabel('Strain')

%-------------------------------------------------------------------------  ---↙
----------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME
% FROM COMPANION EXCEL WORK SHEET 'TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY
    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);
    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')

 %-------------------------------------------------------------------------↙
---------------
% PLOTTING THE TEST DATA AND A PLOT OF THE BASIC STRAIN LIFE LINE AND STAND↙
```

```
ARD DEVIATIONS
% note that the mean 2NF + the % datavar gives the simulated outer bounds o↙
f all test data points,
%
A = xlsread('newtestdatainputs','MATLABINPUTVECTOR');        %NOTE THIS IS ↙
A COLUMN VECTOR
fsc = A(1,1);            %fatigue strength coeff
fdc = A(2,1);            %fatigue ductility coeff
b = A(5,1);              %fatigue strength exponent
c = A(8,1);              %fatigue ductility exponent
E = A(9,1);              % E modulus of elasticity

    NF2 = [10:100:max(NF)];
    stnlife = ((fsc/E).*(NF2).^b)+fdc.*(NF2).^c;
    stnlifestdleft = (NF2-(meandatavar).*NF2);
    stnlifestdrite = (NF2+(meandatavar).*NF2);
    loglog(NF2,stnlife,'r')


end

%
```



```
function F = lifefcn2(x,NF,ex1,ex2,ex3,ex4);
% lifefcn2 - Function to calculate the strain life equation.
% File written by Tom Heffern.  Last modified 18 May 02
% This file was designed to work with strainfit.m
%--------------------------------------------------

F = x(1).*NF.^ex1 + x(2).*NF.^ex2 +x(3).*NF.^ex3 + x(4).*NF.^ex4;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G(1) GA.M


```
%%%  GA.m
%%% driver routine for the genetic computatioanl algorithm STRAIN-LIFE Solu↙
tion
%%% Author:         Ramesh Kolar,  modified by Thomas Heffern  August 23 20↙
02
%%%Affiliations:    Universal Genetictactology

clc
clear all

max_pop = 120;
max_string = 80;   % this revised GA needs a bigger string size
global pop_size str_length max_gen
global pr_cross pr_mute sum_f
global num_mute num_cross generation
global stat_avg stat_max stat_min
global avg_hist max_hist min_hist
%
global Bestchrom
generation = 1;
old_pop = initialize(generation);
while generation <= max_gen-1
    generation = generation + 1;
    generate;
    disp(' It is the generation no. ')
    generation
    staticise(new_pop);
    old_pop = new_pop;
end
genx=1:max_gen+1;
plot(min_hist,'b');
grid on;hold on;
xlabel('generations')
ylabel('total Slope  error')
legend('min history')

% Finding the best solution
[F,I] = sort([new_pop.fitness]);      % sorts the new pop by fitness which ↙
is RMS in ascending order

figure
NF = LOGSPACE(2, 8, 500);

finalstn = (new_pop(I(1)).A.*NF.^new_pop(I(1)).B)+(new_pop(I(1)).C.*(NF).^n↙
ew_pop(I(1)).D)...
    +(new_pop(I(1)).E.*NF.^new_pop(I(1)).F)+(new_pop(I(1)).G.*NF.^new_pop(I↙
(1)).H);
finalstn2 = (new_pop(I(15)).A.*NF.^new_pop(I(15)).B)+(new_pop(I(15)).C.*(NF↙
).^new_pop(I(15)).D)...
```

```
    +(new_pop(I(15)).E.*NF.^new_pop(I(15)).F)+(new_pop(I(15)).G.*NF.^ne
p(I(15)).H);
finalstn3 = (new_pop(I(40)).A.*NF.^new_pop(I(40)).B)+(new_pop(I(40)).C.
).^new_pop(I(40)).D)...
    +(new_pop(I(40)).E.*NF.^new_pop(I(40)).F)+(new_pop(I(40)).G.*NF.^ne
p(I(40)).H);
loglog(NF,finalstn,'b',NF,finalstn2,'k',NF,finalstn3,'r')
% That plotted the 3 best solutions
hold on
%-------------------------------------------------------------------------
----------------------
% PLOT THE ACTUAL TEST DATA POINTS FOR REFERENCE...THESE POINTS COME
% FROM COMPANION EXCEL WORK SHEET 'NEW TEST DATA INPUTS'
% ENSURE THAT THE SPREAD SHEET IS IN THE SAME DIRECTORY

    act2NF = xlsread('newtestdatainputs','2NFvector');
    actstrain = xlsread('newtestdatainputs','Strainvector');
    HCF = xlsread('newtestdatainputs','NAVHCF');
    HCF2NF = HCF(:,3);
    HCFstrn = HCF(:,1);

    loglog(act2NF,actstrain,'k.',HCF2NF,HCFstrn,'r.')


finalize;
```

APPENDIX G(2) INITIALIZE.M


```
function old_pop = initialize(nstart)
% initialization phase
%
global pop_size str_length max_gen
global pr_cross pr_mute sum_f
global num_mute num_cross generation
global stat_avg stat_max stat_min
global avg_hist max_hist

%pop_size= input(' Population Size please...? ');
pop_size=80;                                          % POPULATION SIZE MUCH ↙
MATCH UP TO GENERATE!!!
%str_size = input(' And, Chromo Length - string length ...' );
str_size =240;
str_length = str_size;
max_gen = input(' Now, Maximum number of generations is ...');
pr_cross = input(' And, Cross-over probability is.. about  .4 ');
pr_mute = input(' Finally, Mutational probability is ..needs to be about .0↙
1 for this model ');
%
num_mute = 0;
num_cross = 0;
pr_init=0.5;
for i=1:max_gen,
    avg_hist(i) = 0.0;
    max_hist(i) = 0.0;
    min_hist(i) = 0.0;
end
%% initialize the population
for j=1:pop_size,
    for j1 = 1:str_size,
        old_pop(j).czome(j1) = flip(pr_init);
    end

        [obj_fun_val,A,B,C,D,E,F,G,H] = obj_function(old_pop(j).czome);
        old_pop(j).fitness = obj_fun_val;
        old_pop(j).parent1 = 0;
        old_pop(j).parent2 = 0;
        old_pop(j).xsite = 0;
end
disp(' generation ');
nstart;
staticise(old_pop);
print_init;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G(3) FLIP.M


```
function flip_val = flip(prob)
% returns a value based on a biased flipping of a coin
% 1 or 0 is the outcome
% Uses uniform pdf of computing random number
v1 = rand;
if ( abs(prob -1) <= 1.e-5 )
    flip_val = 1;
elseif (v1 <= prob)
    flip_val = 1;
else
    flip_val =0;
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G(4) OBJ_FUNCTION.M


```
function [obj_fun_val,A,B,C,D,E,F,G,H] = obj_function(x)
 % f(x) = x^n where n=10
 ref_val = 1073741823.0;   % for use with 30 bit chromozomes
 %ref_val = 1023;            % for use with 10 bit partial chroms
 %ref_val = 32767;           % for use with 15 bit partial chroms
 %ref_val = 1048575 ;         %for use with 20 bit
 % THIS REVISED CODE BREAKS THE CHROMOSOMES DOWN INTO PIECES
 %n=10;

 new_str_length = 30;
 A_val = decode(x(1:30),new_str_length);
 B_val = decode(x(31:60),new_str_length);
 C_val = decode(x(61:90),new_str_length);
 D_val = decode(x(91:120),new_str_length);
 E_val = decode(x(121:150),new_str_length);
 F_val = decode(x(151:180),new_str_length);
 G_val = decode(x(181:210),new_str_length);
 H_val = decode(x(211:240),new_str_length);
 % Mapping the large decimal values to the desired ranges


 A = (-5+(75/ref_val)*A_val);

 B = (-4+(4./ref_val)*B_val);

 C = (-15+(25/ref_val)*C_val);

 D = (-1+(1.05/ref_val)*D_val);

 E = (-15+(25/ref_val)*E_val);

 F = (-1+(1.05/ref_val)*F_val);

 G = (-35+(45/ref_val)*G_val);

 H = (-1+(1.8/ref_val)*H_val);


 % computing the revised strain life equation at the various known points
 NF = [138,239.8,565,990,1514,5657,12045,28050,81461,290680,710264,21188202↙
];
 % Known NF vector  corresponding to...
 %Known strain vector
 strain = [.04,.03,.02,.015,.012,.007,.006,.005,.004,.003,.0025,.002] ;
 % The chromosome thinks the answer is....
 stn = (A.*NF.^B)+(C.*NF.^D)+(E.*NF.^F)+(G.*NF.^H);

 slopereal = [(log10(strain(2))-log10(strain(1)))/(log10(NF(2))-log10(NF(1)↙
)),(log10(strain(4))-log10(strain(3)))/(log10(NF(4))-log10(NF(3)))...
```

```
     ,(log10(strain(6))-log10(strain(5)))/(log10(NF(6))-log10(NF(5))),(log1↙
0(strain(12))-log10(strain(11)))/(log10(NF(12))-log10(NF(11)))];
 slopesim = [(log10(stn(2))-log10(stn(1)))/(log10(NF(2))-log10(NF(1))),(log↙
10(stn(4))-log10(stn(3)))/(log10(NF(4))-log10(NF(3)))...
     ,(log10(stn(6))-log10(stn(5)))/(log10(NF(6))-log10(NF(5))),(log10(stn(↙
12))-log10(stn(11)))/(log10(NF(12))-log10(NF(11)))];

 % comparing the answers
 error1 = abs(strain-stn);
 sloperror = abs(slopereal-slopesim)./slopereal;
 sloperror(1) = 1000*sloperror(1);
 sloperror(3) = 1000*sloperror(3);
percent = abs(error1./strain);
percent(6) = 2000*percent(6);
percent(7) = 5*percent(7);
percent(8) = 5*percent(8);
percent(9) = 5*percent(9);
percent(10) = 10*percent(10);
percent(12) = 6000*percent(11);
errorsize = norm(percent)+norm(sloperror);
obj_fun_val = errorsize;
```

APPENDIX G(5) DECODE.M


```
function dec_index = decode(czome,str_length)
% decode string as unsigned binary integer 1.0 t/f
sum = 0;
pof2 = 1;

for j=str_length:-1:1,
    if(czome(j) > 0)
        sum = sum + pof2;
    end
pof2 = pof2 * 2;
end
dec_index = sum;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G(6) GENERATE


```
% file name :       generate.m file
%                   creates new generation
%                   even numbered pop_size
%                   j, mate_1 mate_2 jcross are integers
%                   initialization phase
%  NOTE: THIS GENERATE IS SIGNIFICANTLY DIFFERENT  from previous versions
global pop_size str_length max_gen
global pr_cross pr_mute sum_f
global num_mute num_cross
global stat_avg stat_max stat_min
%added
global child_1 child_2

[F,I] = sort([old_pop.fitness]);      % sorts the old pop by fitness which
is RMS in ascending order
% so the best are I(1), I(2) and I(3)
mate_1 = old_pop(I(1)).czome;
mate_2 = old_pop(I(2)).czome;
lucky = rnd(3, pop_size);        %want two lucky mutants who aren't one of
the best 2  #1
mate_3 = old_pop(I(3)).czome;
lucky = rnd(3, pop_size);        %want two lucky mutants who aren't one of
the best 2  #2
mate_4 = old_pop(I(lucky)).czome;
% I don't know why, but I think the mates are somehow mutating, this is an
attempt to stop that
%KEEP THE TWO BEST
new_pop(1).czome = mate_1;
new_pop(2).czome = mate_2;
%_____________________________________________________________________________
_____________________
% OFFSPRING OF MATE 1 and MATE 2    KING AND QUEEN

for j = 3:2:10
    [child_1 child_2 jcross] = crossover(mate_1,mate_2);
    new_pop(j).czome = child_1;
    new_pop(j+1).czome = child_2;
end

%_____________________________________________________________________________
_____________________
% OFFSPRING OF MATE 3 and MATE 4    THE CLAMPETS

for j = 11:2:50
    lucky = rnd(1, pop_size/2);        %want two lucky mutants  #1
    mate_3 = old_pop(I(lucky)).czome;
    lucky2 = rnd(1, pop_size);        %want two lucky mutants   #2
    mate_4 = old_pop(I(lucky2)).czome;
    %
```

```
    [child_1 child_2 jcross] = crossover(mate_3,mate_4);
    new_pop(j).czome = child_1;
    new_pop(j+1).czome = child_2;
end

%_____________________________________
% OFFSPRING OF MATE 5 and MATE 6    THE NEW RANDOM POP
% NOTE THIS TEMPORARILY MAKES A FRESH "OLD _POP"
pop_size=80;                                        % POPULATION SIZE MUCH MATCH UP TO GENERATE!!!
str_size =240;
pr_init=0.5;
for j=1:pop_size,
    for j1 = 1:str_size,
        old_pop(j).czome(j1) = flip(pr_init);
    end
end

for j = 51:2:80
    lucky = rnd(1, pop_size);        %want two lucky mutants  #1
    mate_3 = old_pop(I(lucky)).czome;
    lucky2 = rnd(1, pop_size);        %want two lucky mutants   #2
    mate_4 = old_pop(I(lucky2)).czome;
    %
    [child_1 child_2 jcross] = crossover(mate_3,mate_4);
    new_pop(j).czome = child_1;
    new_pop(j+1).czome = child_2;
end

%_____________________________________
% Figuring out all the particulars of the chromosomes
for j=1:1:pop_size,

        [obj_fun_val,A,B,C,D,E,F,G,H] = obj_function(new_pop(j).czome);
        new_pop(j).fitness = obj_fun_val;
        new_pop(j).A = A;
        new_pop(j).B = B;
        new_pop(j).C = C;
        new_pop(j).D = D;
        new_pop(j).E = E;
        new_pop(j).F = F;
        new_pop(j).G = G;
        new_pop(j).H = H;
        %new_pop(j).parent1 = mate_1;   %This will not be correct
        %new_pop(j).parent2 = mate_2;    %This will not be the correct parents...don't really care right now
        %new_pop(j).xsite = jcross;   % this doesn't matter much either
```



```
end
```

APPENDIX G(7) CROSSOVER.M


```
function [child_1,child_2,jcross] = crossover(parent1,parent2)
% cross two parent zomes and place in child zomes
% also mutate allele with a probability of pr_mutation
global pop_size str_length max_gen
global pr_cross pr_mute sum_f
global num_mute num_cross
global stat_avg stat_max stat_min
%
% first constant_____________________________________________________________
_____________
if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(1,29);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 30;
end
% ist exchange 1 to 1 and 2 to 2
for j = 1:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 30)
    for j=jcross+1:30,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end

% second constant____________________________________________________________
___________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(31,59);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 60;
end
% ist exchange 1 to 1 and 2 to 2
for j = 31:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
```

```
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 60)
    for j=jcross+1:60,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end

% third constant_____________________________________________________↙
_____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(61,89);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 90;
end
% ist exchange 1 to 1 and 2 to 2
for j = 61:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 90)
    for j=jcross+1:90,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
% fourth constant_____________________________________________________↙
_____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(91,119);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 120;
end
% ist exchange 1 to 1 and 2 to 2
for j = 91:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
```

```
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 120)
    for j=jcross+1:120,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
% fifth constant_____________________________________________________
_____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(121,149);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 150;
end
% ist exchange 1 to 1 and 2 to 2
for j = 121:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 150)
    for j=jcross+1:150,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
% sixth constant_____________________________________________________
_____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(151,179);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 180;
end
% ist exchange 1 to 1 and 2 to 2
for j = 151:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
```

```
if (jcross ~= 180)
    for j=jcross+1:180,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
% seventh constant_____________________________________________________
_____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(181,209);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 210;
end
% ist exchange 1 to 1 and 2 to 2
for j = 181:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 210)
    for j=jcross+1:210,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
% eighth constant______________________________________________________
____________________

if (flip(pr_cross) == 1)
    %do crossover
    jcross = rnd(211,239);
    % cross over between (1,length-1)
    num_cross = num_cross + 1;
else
    jcross = 240;
end
% ist exchange 1 to 1 and 2 to 2
for j = 211:jcross,
    child_1(j) = mutation(parent1(j));
    child_2(j) = mutation(parent2(j));
end
%
% 2nd exchange 1 to 2 and 2 to 1;
if (jcross ~= 240)
```



```
    for j=jcross+1:240,
        child_1(j) = mutation(parent2(j));
        child_2(j) = mutation(parent1(j));
    end
end
```

APPENDIX G(8) MUTATION.M


```
function mute_val = mutation(allele_val)
% simple mutation algorithm
% mutation is logical
global pr_cross pr_mute sum_f
global num_mute num_cross generation
mutate = flip(pr_mute);
%%%%%disp(' in mutation ');
allele_val;
if (mutate == 1)
    num_mute = num_mute+1;
    mute_val = not(allele_val);
else
    mute_val = allele_val;
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G(9) STATICISE


```
function staticise(popln)
global pop_size str_length max_gen
global pr_cross pr_mute sum_f
global num_mute num_cross generation
global stat_avg stat_max stat_min
global avg_hist max_hist min_hist
global generation Bestchrom
%
sum_f = popln(1).fitness;
smin =  popln(1).fitness;
smax =  popln(1).fitness;

for j=2:pop_size,
    fit_loc =  popln(j).fitness;
    sum_f = sum_f + fit_loc;
    if (fit_loc > smax)
        smax = fit_loc;
    elseif (fit_loc <= smin)
        smin = fit_loc;
        Bestchrom = j;
    end
end

[F,II] = sort([popln.fitness]);      % sorts the new pop by fitness which i↙
s RMS in ascending order
min_hist(generation) = popln(II(1)).fitness
disp(' generation ')
generation
savg = sum_f/pop_size;
stat_avg = savg;
stat_max = smax;
stat_min = smin;
%disp(' <<< HIT Carriage Return to continue >>> ')
%pause
%disp (' average ');
savg;
%disp(' max ');
smax;
%disp(' min ');
smin;
%disp(' sum_f ');
sum_f;
avg_hist(generation)=savg;
max_hist(generation)=smax;
%min_hist(generation) = smin;
```

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

1. Bannatine, Julie, A., Comer, Jess, J., Handrock, James, L., Fundamentals of Metal Fatigue Analysis, Prentice-Hall, 1990

2. Weibull, W., *A Statistical Distribution Function of Wide Applicability*, Journal of Applied. Mechanics, 293-297 (1951).

3. Rusk, D.T., Hoffman, P.C., *Developments in Probability-Based Strain-Life Analysis*, 2002 FAA-NASA Aging Aircraft Conference.

4. Annis, Charles, *HCF, g-functions, and Probabilistic Engineering Analysis-A Checklist for Probabilists*, 4th National Turbine Engine High Cycle Fatigue Conference 9-11 February 1999.

5. Leitch, Roger, D., Reliability Analysis for Engineers, Oxford Science Publications, 1995.

6. Massarelli, Peter J., Baber, Thomas T., *Final Report – Fatigue Reliability of Steel Highway Bridge Details*, Virginia Transportation Research Council, Charlottesville, VA, August 2001.

7. Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.

8. Holland, J.H., Adaptation in Natural and Artificial Systems : an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, The University of Michigan Press, 1975.

9. Kolar, Ramesh. Verbal Description of Genetic Algorithms and Sample Genetic Algorithm MATLAB® code provided to student.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
   debarber@nps.navy.mil

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia
   webmaster@tecom.usmc.mil

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia
   ramkeyce@tecom.usmc.mil
   strongka@tecom.usmc.mil
   sanftlebenka@tecom.usmc.mil

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
   doranfv@mctssa.usmc.mil
   palanaj@mctssa.usmc.mil

7. Dr. Paul C. Hoffman, Structures Division
   Code 4.3.3
   Naval Air Systems Command
   Patuxent River, MD 20670-1906
   HoffmanPC@navair.navy.mil
   RuskDT@navair.navy.mil

8. Dr. Ramesh Kolar
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943

9. Dr. E. Roberts Wood
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943

10. Major Thomas Heffern
    22415 Greenview Ct
    Great Mills, MD 20634